GEOMETRIC DISTRIBUTIONAL CONTROL: LEARNING PROGRESS WITH PARTIAL STRUCTURAL KNOWLEDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Partial structural knowledge can constrain admissible decisions without specifying how to achieve task progress. We introduce Geometric Distributional Control (GDC), which formulates decision-making under such knowledge as learning and navigating conditional distributions of feasible action segments. The central principle is positive expected progress of a conditional action-segment distribution over supported horizons, rather than positive progress of every segment. This lets waiting or repositioning coexist with goal-directed behavior and extracts directional knowledge from imperfect trajectories without globally optimal demonstrations. Multi-horizon candidate selection lets the learned guidance respond to the temporal scale of the current context. GDC learns a progress-weighted distribution in a compact latent space and combines its score with available analytical structure to guide online geometric optimization. We establish conditions under which distributional approximation and candidate selection retain positive expected segment progress. A conditional objective-gap bound links representation coverage, value approximation, and online search accuracy to decision quality on a data-supported region. Experiments on traffic control and hierarchically coupled nonconvex merit optimization demonstrate effectiveness across changing constraints and problem scales.

1 INTRODUCTION

Many deployed controllers know enough to reject an action but not enough to choose among the survivors. Actuator limits, collision checks, traffic rules, conservation laws, or lower-level response maps define a context-dependent feasible set; they do not by themselves determine whether the controller should turn left, wait, change basin, or coordinate several levels. Model predictive control (MPC) resolves this ambiguity when a complete model and objective can be optimized online (Mayne et al., 2000; Buerger et al., 2024). Reinforcement learning instead learns a policy or value (Sutton & Barto, 2018; Levine, 2018), but this asks data to represent feasibility, long-horizon ranking, and execution in one object. We study the intermediate information pattern: explicit structure certifies executability, while data must supply only the missing direction.

The central object in GDC is therefore not a sampled policy. It is a *feasible-progress measure* over short action segments. Known structure first restricts the support to an action-segment fiber. A weak signed certificate then applies an exponential change of measure on that same support. The resulting conditional Gibbs measure retains waiting, yielding, reversal, and repositioning modes, but monotonically shifts mass toward larger certified progress. After pushforward to compact latent coordinates, its log-density becomes a data-induced control energy and its score becomes a geometry-aware local direction. This converts short, imperfect feasible snippets into a reusable vector field without claiming a global Bellman value.

The decisive progress condition is distributional: $\mathbb{E}_{\tau\sim q_{\beta,H}(\cdot|c)}[\Delta\rho(\tau,g)] > 0$ for a supported context and horizon, rather than $\Delta\rho(\tau,g) > 0$ for every feasible segment. A vehicle may wait at a red light, or an optimizer may temporarily move away from its eventual basin, while the conditional mixture still advances in expectation. Horizon choice matters because a useful maneuver can have little immediate progress yet positive progress over a longer segment. The progress tilt biases the feasible data toward that expectation without discarding necessary waits.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This measure/control separation differs from diffusion planning and policies (Janner et al., 2022; Chi et al., 2023; Wang et al., 2023): sampling is not itself the executed law. GDC inserts the score into a two-timescale Riemannian energy flow, retracts to the data-supported scaffold, and applies explicit filtering, projection, and fallback. It also differs from offline RL, which learns a reward-optimal policy or value, and from safe RL or shields, where feasibility is often a cost or final repair (Carr et al., 2023; Zhao et al., 2023; Wachi et al., 2024). Here known constraints define the training support, latent scaffold, online energy, candidate checks, and final execution set. Riemannian score modeling (Hyvärinen, 2005; Song & Ermon, 2020; De Bortoli et al., 2022) provides the geometric language, while structured solver maps (Amos & Kolter, 2017; Agrawal et al., 2019; Bertrand et al., 2020) are possible scaffold components rather than the object being learned.

The contribution is a distributional answer to directional ambiguity under partial structural knowledge: seek positive expected signed progress among known-feasible short segments without requiring every segment to improve. The components in Figure 1 serve this criterion. The Gibbs tilt raises mean certified progress while retaining necessary waits and reversals; candidate horizons let that progress be assessed at the temporal scale of the current context. The latent score turns the tilted measure into a local direction, and the geometric flow, explicit checks, projection, and fallback keep online decisions within the supported and known-feasible region. We give conditional descent and execution bounds for this construction and evaluate it in route control and penalized multilevel benchmarks, including transfer with a frozen learned measure and changed known scaffolds. When feasible segments admit a low-dimensional representation ($d_z \ll Hd_u$, Sec. 3), GDC fits a local-direction estimate from short segments without value-function bootstrapping. This representation is an efficiency assumption whose adequacy depends on the coverage and reconstruction quality of the available data.

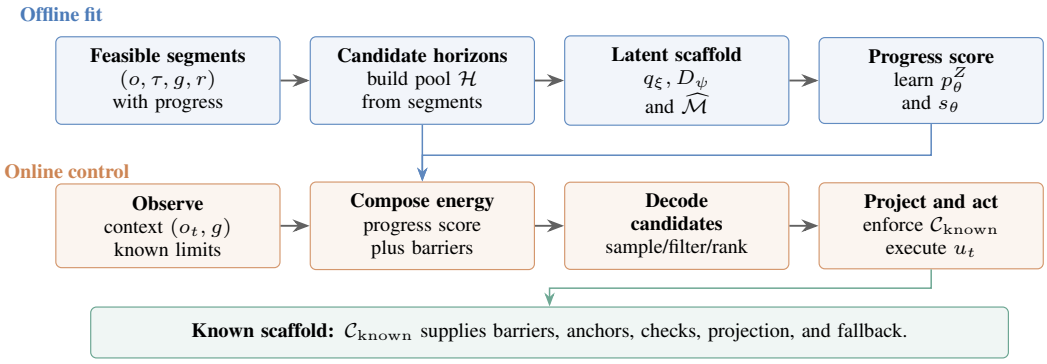


Figure 1: Overview of GDC. Known-feasible segments with weak progress certificates are used offline to build a latent feasible-progress geometry and score field. Online control follows the induced progress-energy direction, while the explicit known scaffold remains responsible for filtering, projection, fallback, and execution safety.

2 Related Work

Dynamic programming ranks actions through a value function, whereas MPC repeatedly solves a specified finite-horizon model (Bellman, 1957; Bertsekas, 2017; Mayne et al., 2000). GDC targets a different information pattern: reliable structure can reject actions, but neither a complete model nor a global value is available to rank the survivors. Offline RL controls extrapolation while learning reward-optimal policies or values (Kumar et al., 2020; Kostrikov et al., 2022); GDC instead learns a weaker local progress direction from short known-feasible segments and treats missing coverage as an abstention event. Score models support trajectory planning and expressive action distributions (Hyvärinen, 2005; Song & Ermon, 2020; Janner et al., 2022; Chi et al., 2023; Wang et al., 2023). In GDC the learned distribution is not the executed policy: its score is one term in a structured latent flow, after which explicit checks, projection, and fallback determine the action. Riemannian score modeling supplies the geometry (De Bortoli et al., 2022); safe RL and shields motivate explicit runtime protection (Achiam et al., 2017; Carr et al., 2023; Zhao et al., 2023); and differentiable solvers may provide known scaffold components (Amos & Kolter, 2017; Agrawal et al., 2019; Bertrand et al., 2020). The distinguishing choice is the division of labor: known structure determines executability, while probability supplies only the missing tangential direction (Appendix A).

3 GEOMETRIC DISTRIBUTIONAL CONTROL

3.1 FEASIBLE ACTION FIBERS, DATA, AND ABSTENTION

Feasible fiber. Let x_t denote the (possibly unobserved) system state, evolving under an unknown or partially known transition f as $x_{t+1} = f(x_t, u_t)$ for action $u_t \in \mathcal{U}$, and observed as $o_t = h(x_t) + \varepsilon_t$ for a fixed goal g and noise ε_t. We distinguish explicit structure $\mathcal{C}_{\text{known}}$ (constraints, geometry, response maps, and projections that can be evaluated online) from residual implicit structure $\mathcal{C}_{\text{implicit}}$ represented statistically. For context $c = (o, g, \mathcal{C}_{\text{known}})$, the explicit model induces the horizon-H feasible fiber

$$\mathcal{F}_H(c) = \{\tau = (u_0, \dots, u_{H-1}) : \tau \text{ satisfies every evaluable constraint in } \mathcal{C}_{\text{known}}\}. \tag{1}$$

Every offline segment τ also carries a scalar *progress certificate* $\Delta\rho(\tau, g) = \rho(x_{t+H}, g) - \rho(x_t, g)$ for some coarse, possibly noisy progress measure ρ: a weak signal of whether τ moved toward g, supplied by the data source rather than learned, and not required to be positive or monotone along the segment. Concretely, in SUMO (Simulation of Urban MObility) $\mathcal{F}_H(c)$ is the set of H-step acceleration/lane-change segments obeying actuator, headway, signal, and collision limits, and $\Delta\rho$ is net route advancement (Sec. 5); in the multilevel problems, a segment is a short joint update of every level's iterate obeying box and step limits, and $\Delta\rho$ is the change in the fixed-weight score Φ. The same fiber and certificate definitions cover both; only $\mathcal{C}_{\text{known}}$ and $\Delta\rho$ change per domain.

Horizon candidates and support. The offline dataset $\mathcal{D} = \{(o_i, \tau_i, g_i, r_i)\}_{i=1}^N$ pairs each segment $\tau_i = (u_{i,t}, \dots, u_{i,t+H_i-1})$ with its certificate $r_i = \Delta\rho(\tau_i, g_i)$. GDC models a finite horizon pool $\mathcal{H}$. At deployment, it refines and filters candidate segments from the available horizons, then ranks the survivors together. The selected horizon is thus context dependent. SUMO uses $\mathcal{H} = \{6, 12, 24\}$ on the 20×20 and transfer maps (Table 1, Figure 3) and $\{6, 24, 48\}$ on the 60×60 maps (Figure 2).

An optional support screen can restrict this pool using local progress certificates and return the known-feasible fallback when no horizon qualifies (Appendix B). The reported SUMO comparison uses the full pool and a runtime candidate-progress gate (Appendix F.1). Below, $H^\star$ denotes the selected candidate's horizon for a fixed context. Zero or negative certificates remain in training when they encode waiting or repositioning.

3.2 POSITIVE EXPECTED PROGRESS AND LATENT GEOMETRY

Gibbs tilt. For a selected horizon, let $q_{\mathcal{D}}^{H^\star}(\cdot \mid c)$ be the empirical measure on $\mathcal{F}_{H^\star}(c)$ and $\bar{r}(\tau, g) = \text{clip}(\Delta\rho(\tau, g), -c_r, c_r)$ the clipped progress certificate. GDC defines the feasible-progress measure by tilting $q_{\mathcal{D}}^{H^\star}$ exponentially toward higher certified progress, with the same inverse-temperature $\beta > 0$ fixed offline:

$$\frac{\mathrm{d}q_{\beta,H^\star}}{\mathrm{d}q_{\mathcal{D}}^{H^\star}}(\tau \mid c) = \frac{\exp[\beta\bar{r}(\tau, g)]}{Z_{\beta,H^\star}(c)}, \quad \operatorname{supp} q_{\beta,H^\star} \subseteq \mathcal{F}_{H^\star}(c), \quad \frac{\partial}{\partial\beta}\mathbb{E}_{q_{\beta,H^\star}}[\bar{r}] = \operatorname{Var}_{q_{\beta,H^\star}}[\bar{r}] \geq 0, \tag{2}$$

where $Z_{\beta,H^\star}(c) = \mathbb{E}_{q_{\mathcal{D}}^{H^\star}}[\exp(\beta\bar{r})]$ normalizes the tilt. The last term shows β monotonically shifts probability mass toward larger certified progress without changing the known-feasible support (middle term); among fiber-supported measures with a given mean progress, this exponential tilt maximizes entropy relative to the empirical base measure. Writing $\mu_{\beta,H}(c) = \mathbb{E}_{q_{\beta,H}}[\bar{r}]$, the variance identity gives

$$\mu_{\beta,H}(c) = \mu_{0,H}(c) + \int_0^\beta \operatorname{Var}_{q_{b,H}}[\bar{r}]\,\mathrm{d}b \geq \mu_{0,H}(c). \tag{3}$$

Thus the tilt cannot reduce mean clipped progress. On a supported context where $\mu_{\beta,H}(c) > 0$, the learned target favors progress in expectation even though individual feasible snippets may wait, yield, or temporarily move backward. Positive expectation is a condition to check or assume for a selected context, not a consequence of feasibility alone; the optional support screen in Appendix B provides an empirical diagnostic. GDC never samples $q_{\beta,H^\star}$ at deployment: it only fixes the training target that is pushed through the latent chart (fit on untilted data) and that a score model is then fit to, below.

Directly searching over a horizon-H action segment lives in a space of dimension Hd_u. When the data admit compression, GDC uses a lower-dimensional $z \in \mathbb{R}^{d_z}$ with $d_z \ll Hd_u$. An encoder

$q_\xi(z \mid \tau, c)$ and decoder $D_\psi(z, c, H)$ first fit these latent coordinates on the *untilted* data, so the coordinates also represent low-progress data-supported snippets:

$$(\hat{\xi}, \hat{\psi}) = \arg\min_{\xi,\psi} \big(\mathcal{L}_{\text{rec}} + \beta_z \mathcal{L}_{\text{KL}} + \lambda_k \mathcal{L}_{\text{known}}\big), \tag{4}$$

where $\mathcal{L}_{\text{rec}}$ is the segment reconstruction error, $\mathcal{L}_{\text{KL}}$ regularizes the latent prior, $\mathcal{L}_{\text{known}}$ penalizes known-constraint violation in decoded segments, and β_z, λ_k are fixed weights. Pushing $q_{\beta,H^\star}$ through the frozen encoder $q_{\hat{\xi}}$ then gives the target latent density

$$q_\beta^Z(z \mid c) = \int q_{\hat{\xi}}(z \mid \tau, c)\, q_{\beta,H^\star}(\tau \mid c)\, \mathrm{d}\tau. \tag{5}$$

GDC does not sample q_β^Z any more than it sampled $q_{\beta,H^\star}$: it fits a denoiser whose implied score approximates $\nabla_z \log q_\beta^Z$, so the online controller can query a direction without running a diffusion sampler. For a clean latent $z_0 \sim q_{\hat{\xi}}(\cdot \mid \tau, c)$ of a training segment τ, standard forward corruption at diffusion step ℓ gives

$$z_\ell = \sqrt{\bar{\alpha}_\ell}\, z_0 + \sqrt{1 - \bar{\alpha}_\ell}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{6}$$

where $\bar{\alpha}_\ell \in (0, 1]$ is the cumulative noise-schedule coefficient at step ℓ. A network F_θ reconstructs the clean latent and its decoded segment,

$$\hat{z}_0 = F_\theta(z_\ell, c, \ell), \qquad \hat{\tau}_0 = D_{\hat{\psi}}(\hat{z}_0, c, H^\star), \tag{7}$$

fit in expectation under the tilted measure by

$$\hat{\theta} = \arg\min_\theta \mathbb{E}_{q_{\mathcal{D}}^{H^\star}}\big[\tilde{w}_\beta\, \ell_{\text{path}}(\theta)\big], \quad \ell_{\text{path}}(\theta) = \|\hat{z}_0 - z_0\|^2 + \lambda_\tau \|\hat{\tau}_0 - \tau\|^2 + \lambda_{\text{anc}} \mathcal{A}_{\text{known}}(\hat{z}_0, \hat{\tau}_0), \tag{8}$$

where $\mathcal{A}_{\text{known}}$ penalizes decoded segments that would violate $\mathcal{C}_{\text{known}}$ and $\lambda_\tau, \lambda_{\text{anc}}$ are fixed weights. Minimizing a denoising loss of this form is the standard route to a score model (Hyvärinen, 2005; Song & Ermon, 2020): the fitted F_θ implicitly defines a density $p_\theta^Z(\cdot \mid c) \approx q_\beta^Z(\cdot \mid c)$ whose score is used online,

$$s_\theta(z, c) = \nabla_z \log p_\theta^Z(z \mid c). \tag{9}$$

The tilt itself enters this training only through the importance weight

$$\tilde{w}_\beta(\tau, c) = \frac{e^{\beta \bar{r}(\tau, g)}}{Z_{\beta,H^\star}(c)} = \frac{\mathrm{d}q_{\beta,H^\star}}{\mathrm{d}q_{\mathcal{D}}^{H^\star}}(\tau \mid c), \tag{10}$$

already introduced in Eq. 2; resampling training segments from $q_{\beta,H^\star}$ directly is an equivalent alternative to reweighting. Either way, Eq. 8 is optimized in expectation under $q_{\beta,H^\star}$, not the raw untilted data, while the unweighted Eq. 4 learns coordinates for the available feasible snippets. Under adequate approximation and coverage, decoding p_θ^Z targets positive expected short-segment progress; it does not assert that every segment improves or that a long rollout reaches the goal. Full objectives, corruption kernels, and anchor definitions are in Appendix C.

Decoder geometry. Geometry is induced by the fitted decoder $D_{\hat{\psi}}$, not the encoder. On the data-supported region $\widehat{\mathcal{M}}^*$, the pullback metric is

$$G(z) = J_{D_{\hat{\psi}}}(z)^\top W_\tau J_{D_{\hat{\psi}}}(z) + \epsilon_G I, \qquad \mathrm{grad}_G\, \Phi(z) = G(z)^{-1} \nabla_z \Phi(z), \tag{11}$$

where $J_{D_{\hat{\psi}}}$ is the decoder Jacobian, $W_\tau \succ 0$ weights action-segment coordinates, and the ridge term $\epsilon_G I$ keeps G invertible where $J_{D_{\hat{\psi}}}$ loses rank; short latent steps then correspond to controlled changes in decoded action segments.

3.3 Two-timescale Riemannian progress control

Online energy. All learned components (the encoder $q_{\hat{\xi}}$, decoder $D_{\hat{\psi}}$, latent density p_θ^Z, and score s_θ) are frozen online; no model is retrained. Given o_t, GDC draws K candidates across the available horizon pool, or across $\mathcal{H}_{\text{supp}}(c)$ when the optional support test is active. For a candidate with horizon H, its local known-feasible set is $\mathcal{M}_{t,H} = \{z \in \widehat{\mathcal{M}}^* : D_{\hat{\psi}}(z, o_t, g, \mathcal{C}_{\text{known}}, H)$ is locally known-feasible$\}$. The energy and flow below are evaluated per

candidate; we write $H_t^\star = H$ and $\mathcal{M}_t = \mathcal{M}_{t,H}$ to simplify notation. The $\epsilon_{\mathrm{s}}^{-1}$ term acts on the fast scaffold-restoration timescale; the remaining terms determine motion within the supported region:

$$\begin{aligned}\Phi_{\mathrm{on,t}}(z) &= \epsilon_{\mathrm{s}}^{-1}\Phi_{\mathrm{scaf}}(z; o_t, \mathcal{C}_{\mathrm{known}}) + \lambda\Phi_{\mathrm{known}}(z; o_t, \mathcal{C}_{\mathrm{known}}) \\ &\quad + (1-\lambda)\Phi_{\mathrm{data}}(z; o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) + \Phi_{\mathrm{task}}(z; o_t, g), \qquad \Phi_{\mathrm{data}} = -\log p_\theta^Z, \\ F_t(z) &= -\operatorname{grad}_G \Phi_{\mathrm{on,t}}(z) \\ &= -G(z)^{-1}\left[\epsilon_{\mathrm{s}}^{-1}\nabla\Phi_{\mathrm{scaf}} + \lambda\nabla\Phi_{\mathrm{known}} - (1-\lambda)s_\theta + \nabla\Phi_{\mathrm{task}}\right].\end{aligned} \tag{12}$$

The experiments use a fixed deterministic controller; λ is a prescribed mixture coefficient, not a tuned online policy. The scaffold term restores candidates to the supported region, the learned score supplies tangential progress, and the known and task terms encode evaluable runtime structure. The fast scaffold term limits drift from the region where the learned score is modeled, while Φ_{data} adds a data-informed direction that the known constraints alone need not supply. Filtering and final projection remain active independently of this mixture; the stochastic and interpolation analyses are deferred to Appendix E.

Guarded step. Let $\Pi_V(v) = v\min\{1, V_{\max}/\|v\|_G\}$ denote the metric velocity cap. One navigation step is the guarded second-order Runge–Kutta (RK2)/retraction update

$$\begin{aligned}&k_1 = \Pi_V(F_t(z_n)), \quad z_{\mathrm{mid}} = z_n \oplus \tfrac{\Delta t}{2}k_1, \quad k_2 = \Pi_V(F_t(z_{\mathrm{mid}})), \quad \bar{z}_{n+1} = z_n \oplus \Delta t\, k_2, \\ &z_{n+1} = (1-\mu_R)\bar{z}_{n+1} + \mu_R\Pi_{\mathcal{M}_t}(\bar{z}_{n+1}).\end{aligned} \tag{13}$$

where $\oplus$ is chart addition or a structured retraction and $\Pi_{\mathcal{M}_t}$ returns the candidate to the supported scaffold. Decoding, feasibility filtering, selection, first-action projection, and fallback are stated once in Algorithm 1; only the accepted first action is executed before replanning. The final projection onto $\mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})$ is a weighted-least-squares $\arg\min_{u\in\mathcal{U}_{\mathrm{known}}}\|u - u_t^{\mathrm{nom}}\|_W^2$, which reduces to clipping for box limits and has a closed form for linear/equality constraints.

Algorithm 1 GDC online progress-energy navigation

1: Observe (o_t, g, u_{t-1}) and form the available horizon pool $\mathcal{H}_t$ (optionally $\mathcal{H}_{\mathrm{supp}}(c)$).
2: **if** $\mathcal{H}_t = \varnothing$ **then**
3: Return the known-feasible fallback action.
4: **end if**
5: Warm-start or sample K candidates across $H \in \mathcal{H}_t$, each locally known-feasible at its own horizon.
6: Form each candidate's $\Phi_{\mathrm{on,t}}$ and $F_t = -\operatorname{grad}_G \Phi_{\mathrm{on,t}}$.
7: **for** $k = 1, \ldots, K$ **do**
8: Take N_r velocity-capped RK2 steps, retracting/backtracking to the data-supported scaffold.
9: Decode $\tau_t^{(k)}$ at its candidate horizon and discard it if it violates $\mathcal{C}_{\mathrm{known}}$.
10: **end for**
11: If no candidate remains, return the known-feasible fallback action.
12: Select the surviving segment minimizing the runtime criterion across horizons and take first action u_t^{nom}.
13: Project u_t^{nom} onto $\mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})$; use the fallback if projection fails.
14: Execute u_t and replan from the next observation.

Thus the repeated cost is $O(KN_r(C_\theta + C_{\mathrm{known}}) + C_{\mathrm{proj}})$ and no model is retrained online. Further retraction and projection details are in Appendix C.3.

Remark 3.1 (Transfer by replacing the known scaffold)**.** *Because only $\mathcal{C}_{\mathrm{known}}$, not the learned $\hat{\theta}, \hat{\xi}, \hat{\psi}$, varies as an argument to Algorithm 1, replacing the known scaffold at deployment is not an additional mechanism: Algorithm 1 is re-evaluated at a new $\mathcal{C}_{\mathrm{known}}^{\mathrm{dep}}$ with parameters frozen, succeeding only within the score model's conditional coverage, detailed in Appendix C.3.*

4 Directional and Execution Guarantees

The claims are local and conditional on the selected, data-supported region.

Assumption 4.1 (Supported positive progress)**.** *For every modeled context, offline segments satisfy $\mathcal{C}_{\text{known}}$. The selected horizon's tilted conditional distribution has $\mathbb{E}_{\tau\sim q_\beta}[\Delta\rho(\tau, g)] > 0$. This is a condition for the directional guarantee, while the optional support test gives an abstention rule when no horizon qualifies.*

Assumption 4.2 (Regular map and execution guard)**.** *The decoder and known-constraint maps are locally Lipschitz on the supported neighborhood. Online navigation uses bounded velocity, retraction or backtracking, decoded candidate filtering, and final projection. The set $\mathcal{U}_{\text{known}}(o_t, u_{t-1})$ is closed and nonempty when projection is invoked, and the fallback belongs to it.*

Theorem 4.3 compares s_θ against the score of an idealized, uncomputed target: $Z_H(z, c)$ is the maximum-entropy partition function over known-feasible H-step *continuations* from z, and $V^{\text{soft}}_{\text{prog}} = -\log Z_H$ its soft value. GDC never evaluates Z_H; the theorem only bounds how far s_θ is from this target's gradient.

Theorem 4.3 (Conditional local descent of the progress-measure score)**.** *Fix a non-abstaining context $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$. Under the maximum-entropy continuation model above, let $Z_H(z, c)$ be the resulting positive C^1 partition function and $V^{\text{soft}}_{\text{prog}}(z, c) = -\log Z_H(z, c)$. If $p^\star(z \mid c) \propto Z_H(z, c)$ and $e_\theta = s_\theta + \nabla V^{\text{soft}}_{\text{prog}}$ satisfies $\|e_\theta\|_{G^{-1}} \le \varepsilon_{\text{approx}}$, then*

$$s_\theta = -\nabla_z V^{\text{soft}}_{\text{prog}} + e_\theta, \qquad \frac{\mathrm{d}}{\mathrm{d}t} V^{\text{soft}}_{\text{prog}}(z_t, c) \le -a_t(a_t - \varepsilon_{\text{approx}}), \quad a_t = \|\operatorname{grad}_G V^{\text{soft}}_{\text{prog}}(z_t, c)\|_G, \tag{14}$$

along the score-only natural-gradient component $\dot z_t = G(z_t)^{-1} s_\theta(z_t, c)$. Hence that component strictly decreases the data-induced soft progress value wherever $a_t > \varepsilon_{\text{approx}}$.

The identity follows from $\log p^\star = -V^{\text{soft}}_{\text{prog}} - \log Z_c$ and Cauchy–Schwarz in the decoder metric. Identifying $p^\star$ with the tilted pushforward q^Z_β of Sec. 3 is a modeling assumption, not a fact derived from the training objective; $\varepsilon_{\text{approx}}$ absorbs both optimization error and any mismatch in that assumption. This licenses using s_θ as a natural-gradient direction rather than a value to be globally maximized: the guarantee is local, not a global Bellman value.

Proposition 4.4 (Per-step scaffold-deviation bound)**.** *Suppose each accepted latent update obeys the stated retraction/backtracking guard, with velocity cap $V_{\max}$, step Δt, and restoration scale ϵ_{s}. Its distance to the current known-feasible latent scaffold $\mathcal{M}_t$ satisfies*

$$d_G(z_t, \mathcal{M}_t) \le \sqrt{\epsilon_{\text{s}}/2}\, V_{\max} \Delta t \tag{15}$$

at each accepted update. This bound does not compare the selected action with an optimal one.

Theorem 4.3 only requires s_θ to approximate an idealized continuation score. Under the stronger assumption that the full online potential (not just its score) tracks a true oracle objective, GDC additionally approaches that objective's constrained optimum as coverage and search improve:

Theorem 4.5 (Approximate optimality on the supported region)**.** *Fix $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$ and a compact, geodesically convex, data-supported region $U_c \subset \widehat{\mathcal{M}}^*$ with an m-strongly geodesically convex oracle objective $\Phi_{\text{opt}}(\cdot, c)$ and unique minimizer $z^\star = \arg\min_{U_c} \Phi_{\text{opt}}(\cdot, c)$. If the online potential tracks the oracle up to a context constant $a(c)$, $\sup_{U_c} |\Phi_{\text{on,t}} - \Phi_{\text{opt}}(\cdot, c) - a(c)| \le \alpha_{\text{val}}$, and online refinement returns $\hat z$ with $\Phi_{\text{on,t}}(\hat z) \le \inf_{U_c} \Phi_{\text{on,t}} + \eta_{\text{opt}}$, then*

$$\Phi_{\text{opt}}(\hat z, c) - \Phi_{\text{opt}}(z^\star, c) \le 2\alpha_{\text{val}} + \eta_{\text{opt}}, \qquad \|u_t - u_t^\star\| \le K_D \sqrt{\tfrac{2(2\alpha_{\text{val}} + \eta_{\text{opt}})}{m}} + \delta_{\text{dec}} + \delta_{\text{proj}}. \tag{16}$$

Here $u_t^\star$ is the oracle first action, K_D the Lipschitz constant of the first-action decoder, and $\delta_{\text{dec}}, \delta_{\text{proj}}$ bound the decoding and final-projection errors.

Corollary 4.6 (Coverage-controlled rate)**.** *If $\alpha_{\text{val}} = O(\varepsilon_n + h_n)$ for n latent training anchors with fill distance h_n and anchor error ε_n, and online refinement after K steps gives $\eta_{\text{opt}} = \eta_K$, then both bounds in Eq. 16 are $O(\varepsilon_n + h_n + \eta_K)$ (resp. its square root): as representation coverage, value approximation, and search accuracy improve, GDC approaches the oracle's first action on U_c. This remains conditional on the value-consistency hypothesis above and on U_c itself; it does not claim optimality outside the data-supported region. Appendix E gives explicit coverage rates.*

Proposition 4.7 (Guarded known-feasible execution)**.** *Under the regular-map and execution-guard assumption, every action returned by Algorithm 1 satisfies $u_t \in \mathcal{U}_{\text{known}}(o_t, u_{t-1})$.*

The result follows directly from final projection or the assumed feasible fallback. Appendices D–E contain all proofs, the coverage-rate details for specific anchor-placement regimes, and additional Langevin and Wasserstein results. None of these results assert optimality outside the data-supported region U_c or unknown implicit feasibility outside the supported scaffold.

5 EXPERIMENTS

We test the same role separation in two domains. SUMO uses explicit traffic and actuator checks with learned route progress; multilevel optimization uses analytic objectives, gradients, projections, and local relaxation updates with learned cross-level progress. **Headline results.** On the complex SUMO map, GDC arrives on every route with zero collisions, a 9–49 point arrival gain over rule, behavior cloning (BC), PPO, and SAC; on held-out bilevel and trilevel instances it reaches normalized gaps below 0.65% while satisfying every box and step constraint. Appendix F contains the domain equations, protocols, benchmark definitions, and all secondary results.

5.1 SUMO ROUTE-PROGRESS CONTROL

Task and information. The ego vehicle replans at each 0.5 s SUMO simulation step from current localization, route, lane, leader, signal, and speed features; no controller receives future traffic trajectories (Lopez et al., 2018; Krajzewicz et al., 2012). Actions are acceleration and lane change. The known scaffold checks actuator, headway, signal, and collision constraints and supplies the fallback; weak rollouts provide route-progress certificates. Evaluation uses 100 closed-loop episodes per row except sparse-map SAC (90). The complex map is a perturbed 20×20 two-lane grid with 18 lights, 64 slow detours, and 76 bottlenecks. Constraint equations and limits are in Appendix F.1.

Table 1: SUMO closed-loop results across the sparse and complex maps. All rows have zero mean collisions. "Viol." is the dimensionless aggregate residual returned by the known-constraint checker per decision, with zero denoting no residual beyond its numerical tolerance.

	Sparse				Complex			
Method	Prog.	Arr.	Viol.	Speed	Prog.	Arr.	Viol.	Speed (m/s)
GDC fixed-H	1.000	1.00	1.48×10^{-7}	10.97	1.000	1.00	1.59×10^{-5}	9.25
GDC multi-H	**1.000**	**1.00**	**0.00**	10.58	**1.000**	**1.00**	**0.00**	8.96
Rule safety	1.000	.99	1.33×10^{-1}	11.05	0.989	.91	1.48×10^{-1}	9.00
Weak behavior	0.962	.84	1.44×10^{-2}	7.86	0.841	.55	1.44×10^{-2}	5.65
Behavior cloning	0.931	.76	3.55×10^{-3}	6.26	0.770	.51	3.90×10^{-3}	4.74
PPO	0.997	.97	1.27×10^{-2}	7.87	0.965	.81	1.29×10^{-2}	8.36
SAC	1.000	1.00	1.82×10^{-2}	10.72	0.960	.79	2.65×10^{-2}	8.51

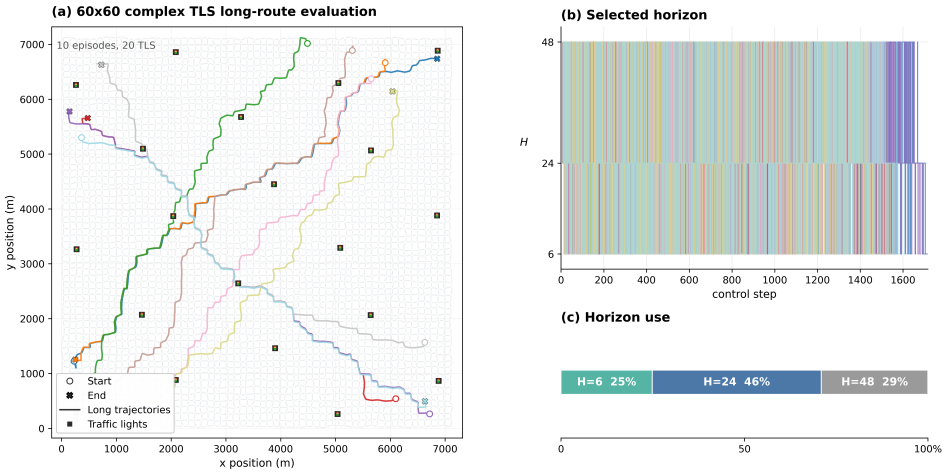


Figure 2: 60×60 long-route stress test. All ten 7.0–$8.1\,\mathrm{km}$ routes reach their goals in $1{,}475$–$1{,}717$ control steps with zero collisions and no timeout; $H = 24$ or 48 is selected on 75.2% of decisions.

Results. GDC reaches every route on both maps with no collisions (Table 1). On the complex map, multi-H improves arrival over Rule, BC, PPO, and SAC by 9–49 percentage points; relative to fixed-

H, it preserves arrival and changes the mean known residual from 1.59×10^{-5} to zero. The two GDC rows share the same data, filter, and fallback, focusing the comparison on temporal candidate pooling. Across additional test-seed batches with the checkpoints frozen, multi-H maintains arrival 1.00 ± 0.00 with zero recorded collisions and known-constraint residuals; fixed $H = 12$ has arrival 0.99 ± 0.02 (Appendix Table 7). Appendix Figure 5 and Figure 2 show adaptive temporal support from the $1.15\,\mathrm{km}$ complex map to the $7.4\,\mathrm{km}$ long-route stress test; an intermediate 60×60 map (10 routes, mean $1.72\,\mathrm{km}$) reaches the same 1.00 arrival with zero collisions and 91.4% long-horizon use (full breakdown in Appendix F.2). This is the operational role of positive expected segment progress: a stop at a signal need not advance the vehicle immediately, but it can be part of a goal-reaching conditional segment distribution. The selected shorter horizons appear near local traffic and constraint-sensitive events, while $H = 24$ or 48 accounts for 75.2% of decisions on the $7.4\,\mathrm{km}$ long routes, all of which finish without collision.

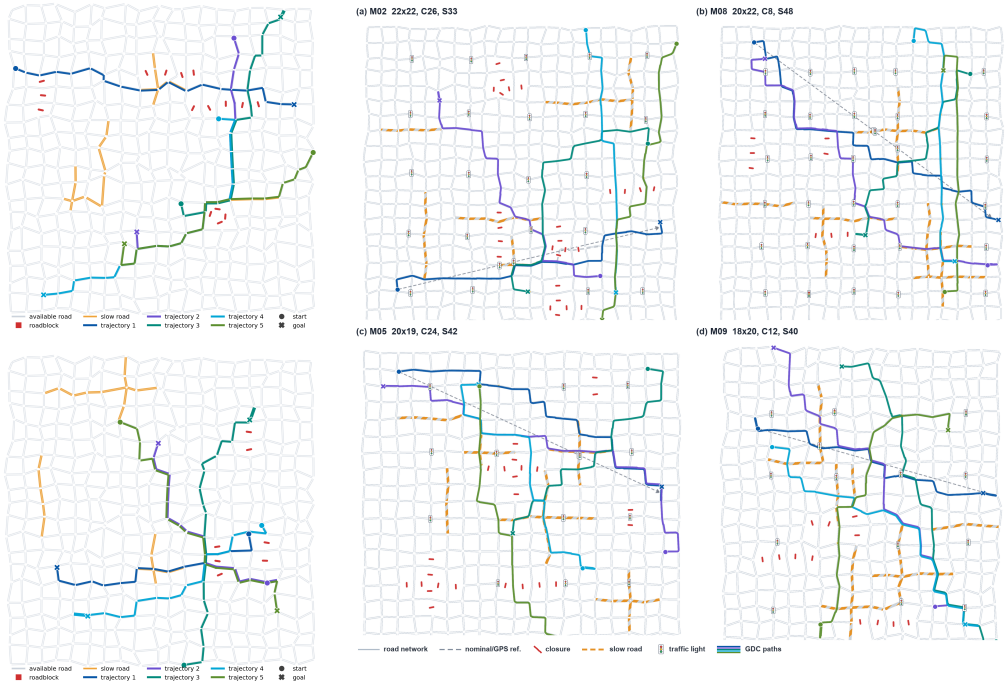


Figure 3: Transfer under changed known scaffolds. Left: two example maps from the 100-map held-out blocked-map benchmark, sharing the base topology but with different closures and slow roads; each shows five sampled routes. Right: held-out graphs with changed dimensions, geometry, traffic parameters, and bottlenecks. The progress model is frozen; the deployment map enters only through $\mathcal{C}_{\mathrm{known}}$, so this is transfer of the learned direction, not map inference.

The frozen score also completes all transfer routes (Figure 3): on 100 blocked-map evaluations, arrival/progress reach 1.00 with zero collisions and non-fallback (learned) control on 99.7% of decisions; on 20 graph-and-disturbance traces, arrival/progress reach 1.00 with zero collisions. Transfer settings and additional maps are in Appendix F.2.

5.2 Parameterized hierarchical merit optimization

Task and information. The second domain uses penalized joint objectives derived from parameterized nonconvex quartic-well bilevel and trilevel problems. Every online method receives the current iterate, held-out descriptor ϑ, analytic objectives and gradients, local relaxation, and identical box/step projections; none receives the global solution, offline multistart reference, or SCIP (Solving Constraint Integer Programs) incumbent. GDC learns a joint cross-level direction from short weak-solver segments. We run at most 260 bilevel or 252 trilevel iterations and report the common fixed-weight score Φ and normalized multistart-reference gap. Problem formulations, solver budgets, and timing data are in Appendix F.3. The scalar Φ measures penalized joint quality.

Table 2: Held-out multilevel evaluation. Known, Nested, and GDC are online methods with the same analytic scaffold. Scores use the same fixed-weight Φ (lower is better). Ref. Φ is the offline multistart reference; Gap (%) is GDC's normalized gap to it. All reported GDC iterates satisfy the explicit box, step, and known-feasibility constraints. SCIP/GDC time is the wall-time factor between SCIP branch-and-cut and GDC on the same instance; "$>$" marks SCIP runs that reached the four-hour limit without certifying optimality.

	Bilevel				Trilevel			
	d=6	d=8	d=10	d=12	d=4	d=6	d=8	d=10
Known	0.7271	0.6208	0.5421	1.0055	0.5730	1.0206	0.9517	1.0590
Nested	0.6998	0.6017	0.5089	0.9147	0.6584	1.1448	1.0723	1.1460
GDC	**0.1241**	**0.0881**	**0.1215**	**0.1345**	**0.1687**	**0.1801**	**0.1312**	**0.1813**
Ref. Φ	0.1228	0.0857	0.1187	0.1323	0.1648	0.1787	0.1228	0.1785
Gap (%)	0.096	0.194	0.229	0.179	0.345	0.093	0.640	0.187
SCIP/GDC time	0.5×	$>$ 249×	$>$ 275×	$>$ 173×	4.1×	$>$ 123×	$>$ 103×	$>$ 104×

Results. GDC reduces Φ by 70.6–86.6% relative to Known on all eight held-out instances (Table 2) under the same scaffold; every normalized gap is below 0.65%, and every reported iterate satisfies the explicit box and step constraints. Figure 4 shows that most improvement occurs early, followed by stable refinement toward the offline incumbent. GDC completes online evaluation in 44.7–138.9 s across all instances, versus a four-hour SCIP budget; six of eight SCIP runs reach that cap, while the two completed runs take 22.6 and 327.8 s (per-instance wall time and the resulting speed factors are in Appendix F.3).

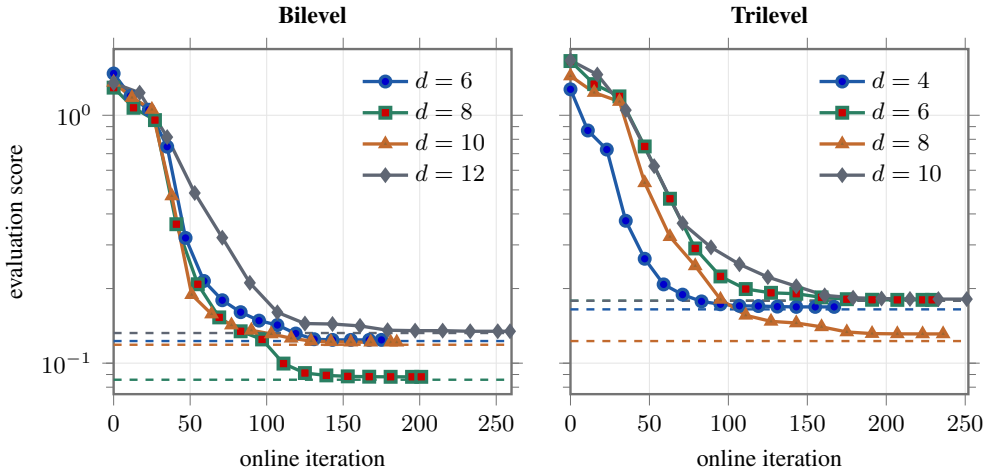


Figure 4: Online convergence of GDC against SCIP branch-and-cut incumbents. Solid curves show GDC over online iterations; matched dashed lines are offline SCIP incumbents, not online competitors. GDC approaches the reference basin across both families.

6 Conclusion

GDC centers control on positive expected progress over a conditional distribution of known-feasible segments, while allowing individual segments to wait, yield, or temporarily regress. Progress weighting and horizon selection define this directional target; the learned score, geometric flow, and known-constraint guard turn it into online actions. Under the stated approximation and convexity conditions, the learned score satisfies a local descent bound and the guard bounds per-step scaffold deviation while preserving known-feasible execution regardless of the score's accuracy. Across SUMO route control and hierarchical merit benchmarks, the frozen multi-H controller reaches every route in the reported test-seed batches with zero collisions, and the penalized-merit normalized gaps stay below 0.65%, improving on rule, imitation, and reinforcement-learning baselines while reusing the frozen progress model under changed scaffolds. Its principal limitation is structural: the efficiency gain relies on an accurate, well-covered compact representation of feasible segments. Performance outside that region is not characterized by the present experiments or guarantees.

Reproducibility Statement

The paper specifies the offline/online decomposition, optional support screen, execution guards, evaluation counts, information conditions, and primary metrics in the main text. The appendices provide training-data scales, complete loss functions, horizon pooling, integration and projection rules, domain equations, benchmark construction, all secondary tables, diagnostic figures, and proofs. No identifying repository link is included in the anonymous submission.

AI Use Statement

Generative AI tools were used to aid and polish writing, for literature retrieval and discovery of related work, and to assist with proving mathematical claims. The authors reviewed the AI-assisted text, mathematical claims and proofs, and citations, and take responsibility for the final content of this work.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31, 2017.

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9558–9570, 2019.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *International Conference on Learning Representations*, 2023.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *Proceedings of the 18th European Control Conference*, pp. 3420–3431, 2019. doi: 10.23919/ECC.2019.8796030.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 136–145, 2017.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: On the curvature of deep generative models. In *International Conference on Learning Representations*, 2018.

Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations*. Birkhäuser, Boston, 1997. doi: 10.1007/978-0-8176-4755-1.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

Quentin Bertrand, Quentin Klopfenstein, Mathieu Blondel, Samuel Vaiter, Alexandre Gramfort, and Joseph Salmon. Implicit differentiation of Lasso-type models for hyperparameter optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 810–821, 2020.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Volume I*. Athena Scientific, 4 edition, 2017.

Johannes Buerger, Mark Cannon, and Martin Doff-Sotta. Safe learning in nonlinear model predictive control. In *Proceedings of the 6th Annual Learning for Dynamics and Control Conference*, volume 242 of *Proceedings of Machine Learning Research*, pp. 603–614, 2024.

Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. Safe reinforcement learning via shielding under partial observability. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37 (12):14748–14756, 2023. doi: 10.1609/aaai.v37i12.26723.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations*, 2023.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Jacob K. Christopher, Stephen Baek, and Ferdinando Fioretto. Constrained synthesis with projected diffusion models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007. doi: 10.1007/s10479-007-0176-2.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2406–2422, 2022. doi: 10.52202/068431-0175.

Stephan Dempe and Alain Zemkoho (eds.). *Bilevel Optimization: Advances and Next Challenges*, volume 161 of *Springer Optimization and Its Applications*. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-52119-6.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915, 2022.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker–Planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998. doi: 10.1137/S0036141096303359.

Bettina Könighofer, Roderick Bloem, Nils Jansen, Sebastian Junges, and Stefan Pranger. Shields for safe reinforcement learning. *Communications of the ACM*, 68(11):80–90, 2025. doi: 10.1145/3715958.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO—Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3–4):128–138, 2012.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using SUMO. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems*, pp. 2575–2582, 2018. doi: 10.1109/ITSC.2018.8569938.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22825–22855, 2023.

David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000. doi: 10.1016/S0005-1098(99)00214-9.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 745–750, 2007. doi: 10.1145/1273496.1273590.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 24(1):1607–1612, 2010. doi: 10.1609/aaai.v24i1.7727.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Tong Su, Tong Wu, Junbo Zhao, Anna Scaglione, and Le Xie. A review of safe reinforcement learning methods for modern power systems. *Proceedings of the IEEE*, 113(3):213–255, 2025. doi: 10.1109/JPROC.2025.3584656.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. doi: 10.1162/NECO_a_00142.

Akifumi Wachi, Xun Shen, and Yanan Sui. A survey of constraint formulations in safe reinforcement learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8262–8271, 2024. doi: 10.24963/ijcai.2024/913.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023.

Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, and Daniela Rus. SafeDiffuser: Safe planning with diffusion probabilistic models. *arXiv preprint arXiv:2306.00148*, 2023.

Weiye Zhao, Tairan He, Rui Chen, Tianhao Wei, and Changliu Liu. State-wise safe reinforcement learning: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 6814–6822, 2023. doi: 10.24963/ijcai.2023/763.

Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.

A Extended Related Work

This appendix expands the related work summarized in Section 2. GDC is a methodology for converting feasible-progress data into an executable geometric control field, so related work is organized by methodological role rather than by application domain.

Value functions, predictive control, and partial structure. Dynamic programming and optimal control use value functions to rank actions by their future consequences (Bellman, 1957; Bertsekas, 2017; Bardi & Capuzzo-Dolcetta, 1997). Model-predictive control uses a specified dynamics model, objective, and horizon to compute a local action by online constrained optimization (Mayne et al., 2000; Buerger et al., 2024). GDC starts from a different information pattern: the controller has reliable partial structure that can rule out infeasible actions, but it does not have a complete model or Bellman value that ranks the remaining feasible directions. The methodological target is therefore a local progress-value gradient on a known-feasible scaffold, not a full cost-to-go or a complete online planner.

Dataset-supported value learning. Offline reinforcement learning learns policies or value functions from fixed datasets while controlling distribution shift. Conservative and implicit value methods avoid overly optimistic extrapolation away from the dataset support (Kumar et al., 2020; Kostrikov et al., 2022). GDC shares the concern that learned decisions should remain data-supported, but it weakens the learning target. The dataset need only contain known-feasible local segments with signed progress certificates; it need not contain expert rollouts or enough information to identify a reward-optimal policy. The learned object is a progress energy whose score chooses among feasible local motions, and the execution layer still enforces explicit constraints.

Score-based distributions as control energies. Score matching and score-based generative modeling learn gradients of log densities (Hyvärinen, 2005; Vincent, 2011; Song & Ermon, 2020; Ho et al., 2020). In decision-making settings, diffusion models have been used as trajectory planners or expressive policy classes: Diffuser performs planning through guided denoising (Janner et al., 2022), Diffusion Policy samples actions from a conditional denoising process (Chi et al., 2023), and diffusion policies have also been used in offline RL or fine-tuned with policy-gradient objectives (Wang et al., 2023; Zhu et al., 2023; Ren et al., 2025). GDC uses the same statistical object, a learned conditional density and its score, with a different interface to control. The distribution is not the policy to execute. Its score is interpreted as a progress-energy gradient and inserted into a structured latent vector field together with known-constraint and task potentials.

Exponential tilting and guided diffusion. The Gibbs tilt in (2) is the same exponential change of measure that underlies reward-weighted regression, relative-entropy policy search, and advantage-weighted regression (Peters & Schaal, 2007; Peters et al., 2010; Peng et al., 2019). Diffusion-based offline RL methods likewise target a behavior distribution reweighted by an exponentiated critic, either by resampling diffusion samples (Chen et al., 2023; Hansen-Estruch et al., 2023) or by learning an exact energy guidance (Lu et al., 2023). Guided diffusion adds the gradient of a classifier, return model, or cost to the learned score during reverse-time sampling (Dhariwal & Nichol, 2021; Ho & Salimans, 2022; Janner et al., 2022); Decision Diffuser instead conditions a diffusion model on returns or constraints and recovers actions with an inverse-dynamics model (Ajay et al., 2023); and constraint-aware samplers insert barrier or projection steps into the reverse process (Xiao et al., 2023; Christopher et al., 2024). We do not claim that exponential tilting, or adding other potentials to a learned score, is new. GDC differs in how the tilted score is used. (i) The tilt weight is a clipped, short-horizon progress label supplied with each known-feasible segment, not a return, advantage, or critic value estimated by bootstrapping, and the tilt acts on the known-feasible fiber rather than on unconstrained behavior data. (ii) The tilted density is fit on a latent chart of action segments and is not sampled at deployment; its score is one term of a velocity-capped, retracted Riemannian descent flow together with known-constraint and task potentials, rather than the drift of a reverse-time sampler or a policy fitted by weighted regression. (iii) Executability is not delegated to guidance or to a soft penalty: decoded candidates are filtered against $\mathcal{C}_{\mathrm{known}}$, the executed action is projected onto the known-feasible set, and a known-feasible fallback is used otherwise, so the feasibility statement of Proposition 4.7 does not depend on the accuracy of the learned score.

Geometry of learned distributions. Riemannian score-based modeling extends score learning to manifold-supported distributions (De Bortoli et al., 2022), and decoder-pullback metrics describe the geometry of latent spaces of deep generative models (Arvanitidis et al., 2018); GDC uses such a metric online. Variational formulations of the Fokker–Planck equation view diffusion dynamics as Wasserstein gradient flows of free energy (Jordan et al., 1998). Neural differential equations and physics-informed learning show complementary ways to encode continuous dynamics or physical residuals in learned models (Chen et al., 2018; Raissi et al., 2019). GDC uses these geometric ideas for a control-specific purpose: partial structure defines the latent scaffold, metric, potentials, and projection maps on which the learned score is allowed to act. The geometry is not only a generative modeling device; it is the interface that makes the progress score composable with explicit feasibility constraints.

Explicit feasibility layers. Safe and constrained learning methods encode safety as constraints on expected costs, state-wise conditions, or policy updates (Achiam et al., 2017; Zhao et al., 2023; Wachi et al., 2024; Su et al., 2025). Runtime shields instead filter or repair actions proposed by a learned policy (Carr et al., 2023; Könighofer et al., 2025), and control barrier functions act as safety filters when the relevant constraint dynamics are known (Ames et al., 2019). GDC follows the broader principle that feasibility should not be left entirely to a black-box policy, but changes where the constraints enter the method. Known constraints define the training support, latent scaffold, online potentials, candidate filters, and final projection. Thus feasibility is not merely a penalty or a post-hoc correction; it is the coordinate system in which the learned progress direction is represented.

Structured solvers as known scaffolds. Differentiable optimization layers and implicit-differentiation methods expose the solution maps of specified optimization problems to learning systems (Amos & Kolter, 2017; Agrawal et al., 2019; Bertrand et al., 2020). These methods are most direct when the relevant optimization problem and its solution map are the object to differentiate through. GDC uses structured solvers more abstractly: a response map, projection, conservation law, or constraint checker can be part of $\mathcal{C}_{\text{known}}$, while the missing object remains the progress direction over the resulting feasible scaffold. This keeps the methodology independent of any one structured application.

B Extended Definitions and Problem Formulation

B.1 Notation

Scalars are written in italic lowercase letters, vectors in bold lowercase letters when their component structure matters, and sets in calligraphic uppercase letters. The system state, observation, control input, and goal at time t are denoted by $x_t \in \mathcal{X}$, $o_t \in \mathcal{O}$, $u_t \in \mathcal{U}$, and $g \in \mathcal{G}$, respectively. A short action segment is $\tau_t = (u_t, \dots, u_{t+H-1})$, and its signed progress certificate is $\Delta\rho(\tau_t, g)$. The explicit known constraint set is $\mathcal{C}_{\text{known}}$, while $\mathcal{C}_{\text{implicit}}$ denotes residual implicit feasibility and preference structure represented statistically from data. The reported controller ranks candidates from a finite horizon pool; $H^\star$ denotes the horizon of a selected candidate. In the optional support-aware variant, an empty admissible pool invokes the fallback. The progress-tilted empirical segment distribution at tilt parameter β is denoted by $q_\beta(\tau \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$.

Latent action-segment coordinates are denoted by $z \in \mathcal{Z}$. The empirical feasible-progress map is $\widehat{\mathcal{M}}^* \subseteq \mathcal{Z}$, with encoder/posterior $q_\xi(z \mid \tau, o, g, \mathcal{C}_{\text{known}})$ and horizon-conditioned decoder $D_\psi(z, o, g, \mathcal{C}_{\text{known}}, H^\star)$. The learned progress-tilted latent density is $p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$, its score is $s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) = \nabla_z \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$, and $G(z)$ denotes the decoder-pullback metric used for Riemannian gradients. When decoded segment or trajectory densities appear, we write p_θ^τ; superscripts are omitted only where the density domain is unambiguous. We write $p_\theta^\tau(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$ for the segment law decoded at a latent point z and $p_\theta^\tau(\cdot \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$ for its marginal under $p_\theta^Z(\cdot \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$. We write $\|\cdot\|$ for the Euclidean norm unless a metric subscript is shown.

B.2 PROBLEM SETTING AND DATA

Explicit and implicit physics. Consider a discrete-time dynamical system

$$x_{t+1} = f(x_t, u_t), \tag{17}$$

where $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ is the system state and $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. The controller has access to partial observations

$$o_t = h(x_t) + \varepsilon_t, \tag{18}$$

where $h : \mathcal{X} \to \mathcal{O}$ is the (possibly nonlinear) observation map and ε_t is measurement noise.

GDC assumes that some physical structure is explicit enough to be enforced online. This includes known constraints, actuator limits, geometric boundaries, conservation laws, or low-order dynamics that can be evaluated reliably. The remaining structure is treated as *implicit physics*: it may be unknown, but it may also be known in principle and simply too complex, coupled, high-dimensional, or expensive to model and optimize online. We write this bookkeeping distinction as

$$f = f_{\text{explicit}} + f_{\text{implicit}}, \qquad \mathcal{C} = \mathcal{C}_{\text{known}} \cap \mathcal{C}_{\text{implicit}}, \tag{19}$$

where f_{explicit} and $\mathcal{C}_{\text{known}}$ are the parts handled by explicit geometry, barriers, anchors, or filters, while f_{implicit} and $\mathcal{C}_{\text{implicit}}$ denote residual dynamics, contact effects, cross-layer couplings, latent preferences, and feasibility patterns that are learned statistically from data. This decomposition is not required to be a physically identifiable additive split; it specifies what is enforced analytically and what is represented through the learned progress distribution.

Feasible trajectory data with progress certificates. We assume access to offline known-feasible trajectories from which short segments can be extracted. After the horizon-selection procedure below, the training set is written as

$$\mathcal{D} = \{(o_i, u_i, \tau_i, g_i, r_i)\}_{i=1}^{N}, \tag{20}$$

where $\tau_i = (u_{i,t}, \dots, u_{i,t+H_i-1})$ is a selected horizon-H_i trajectory segment containing H_i controls, g_i is a goal or task specification, and r_i is a weak progress certificate. We do not assume that the trajectory segment reaches the final goal, nor that every segment has positive progress. Instead, each segment is known-feasible with respect to $\mathcal{C}_{\text{known}}$ and carries a signed certificate such as

$$r_i = \Delta\rho(\tau_i, g_i) = \rho(x_{i,t+H_i}, g_i) - \rho(x_{i,t}, g_i), \tag{21}$$

for a coarse progress measure ρ. Positive r_i indicates local improvement, while negative or near-zero r_i may still correspond to necessary waiting, backing up, repositioning, or safety-preserving behavior. The training objective uses these certificates to bias the learned conditional distribution so that progress is positive in expectation over short trajectories under the data-supported contexts:

$$\mathbb{E}_{\tau \sim p_{\mathcal{D}}(\cdot \mid o, g, \mathcal{C}_{\text{known}})}[\Delta\rho(\tau, g)] > 0, \tag{22}$$

rather than imposing monotone progress on every individual snippet. No dense reward or complete success labels are required.

Candidate horizons and optional support test. The online controller can rank candidates from a finite horizon pool $\mathcal{H} = \{H_1, \dots, H_M\}$. A conservative support-aware option first screens this pool using a context neighborhood $\mathcal{N}(o, g, \mathcal{C}_{\text{known}})$. For each $H \in \mathcal{H}$, cut all known-feasible windows $\tau_i^H = (u_{i,t}, \dots, u_{i,t+H-1})$ in the neighborhood and compute $r_i^H = \Delta\rho(\tau_i^H, g_i) = \rho(x_{i,t+H}, g_i) - \rho(x_{i,t}, g_i)$. Define the clipped progress weight

$$w_i^H = \exp\big(\beta \,\mathrm{clip}(r_i^H, -c, c)\big), \tag{23}$$

and the weighted empirical progress

$$\hat{\mu}_H(o, g, \mathcal{C}_{\text{known}}) = \frac{\sum_{i \in \mathcal{N}(o,g,\mathcal{C}_{\text{known}})} w_i^H r_i^H}{\sum_{i \in \mathcal{N}(o,g,\mathcal{C}_{\text{known}})} w_i^H}. \tag{24}$$

Let $\hat{\sigma}_H$ be the corresponding weighted standard deviation, N_H the number of feasible windows, and

$$N_H^{\text{eff}}(o, g, \mathcal{C}_{\text{known}}) = \frac{\left(\sum_{i \in \mathcal{N}(o,g,\mathcal{C}_{\text{known}})} w_i^H\right)^2}{\sum_{i \in \mathcal{N}(o,g,\mathcal{C}_{\text{known}})} (w_i^H)^2} \tag{25}$$

the effective sample size of the tilted neighborhood. Candidate horizons with no feasible windows in the neighborhood are treated as inadmissible. In the support-aware option, define the empirical screening statistic

$$\widehat{\mathrm{LCB}}_H(o, g, \mathcal{C}_{\mathrm{known}}) = \hat{\mu}_H(o, g, \mathcal{C}_{\mathrm{known}}) - \alpha_{\mathrm{conf}} \frac{\hat{\sigma}_H(o, g, \mathcal{C}_{\mathrm{known}})}{\sqrt{N_H^{\mathrm{eff}}(o, g, \mathcal{C}_{\mathrm{known}})}}. \tag{26}$$

The shortest supported horizon is

$$H^\star(o, g, \mathcal{C}_{\mathrm{known}}) = \min \left\{ H \in \mathcal{H} : \widehat{\mathrm{LCB}}_H(o, g, \mathcal{C}_{\mathrm{known}}) > \delta_{\mathrm{prog}},\ N_H^{\mathrm{eff}}(o, g, \mathcal{C}_{\mathrm{known}}) \geq N_{\min} \right\}. \tag{27}$$

This is a heuristic support screen: overlapping trajectory windows and weighted sampling do not make $\widehat{\mathrm{LCB}}_H$ a calibrated confidence bound. Thus $H^\star$ is a starting horizon for the support-aware option. The reported SUMO experiments instead rank candidates from their finite horizon pool; the horizon is chosen with the final candidate.

If the set in (27) is empty, the support-aware option does not use learned progress in that context:

$$H^\star(o, g, \mathcal{C}_{\mathrm{known}}) = \varnothing. \tag{28}$$

Such data may still train the feasible action-segment map, because it describes what can be executed safely, but the progress tilt is disabled or downweighted for that context. Online control then falls back to the known-feasible controller or sets $\lambda \to 1$ in the physics–data interpolation. This option avoids using a learned direction in a neighborhood with insufficient empirical support. In the notation below, $H^\star$ denotes the fixed horizon of the candidate being analyzed, or the shortest supported horizon in the conservative option, and is treated as part of the conditioning context unless it matters for the algorithm.

Control objective. The goal is to design a feedback controller $u_t = \pi(o_t, g)$ that produces safe, task-consistent, and *directionally correct* actions by leveraging both the geometric structure of $\widehat{\mathcal{M}}^*$ and the distributional knowledge encoded in $\mathcal{D}$.

Formally, let $\mathcal{C} \subset \mathcal{U}$ denote the full safety constraint set and $\ell : \mathcal{O} \times \mathcal{U} \to \mathbb{R}$ a task cost. The ideal constrained feedback objective is

$$\min_{\pi}\ \mathbb{E}\left[\sum_{t=0}^{T} \ell(o_t, u_t)\right] \quad \text{subject to} \quad u_t \in \mathcal{C}\ \ \forall t. \tag{29}$$

GDC addresses this by enforcing $\mathcal{C}_{\mathrm{known}}$ explicitly while learning a data potential whose score represents the implicit physics, coupled feasibility patterns, and goal-conditioned progress information captured by feasible trajectory demonstrations with progress certificates. Consequently, the hard execution guarantee is stated for $\mathcal{C}_{\mathrm{known}}$; membership in $\mathcal{C}_{\mathrm{implicit}}$ is a data-supported statistical property rather than an analytic certificate outside the modeled support.

C Training and Deployment Details

C.1 Overview and offline construction

GDC has an offline construction stage and an online navigation stage. The term *Geometric Distributional Control* refers to the complete framework, while *GDC online control* refers to the frozen feedback controller deployed after offline learning. The two stages are:

Stage	Inputs	Outputs / actions
Offline learning	Known-feasible trajectories, progress certificates, goals, observations, and $\mathcal{C}_{\text{known}}$	Candidate horizon pool $\mathcal{H}$, posterior model $q_\xi(z \mid \tau, o, g, \mathcal{C}_{\text{known}})$, decoder D_ψ or likelihood p_ψ, progress path $p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$, and score s_θ
Online control	Current observation o_t, goal g, known constraints $\mathcal{C}_{\text{known}}$, and fixed offline models	Sample/refine latent candidates z across $\mathcal{H}$, decode candidate action segments, filter and rank by known constraints, execute the first action u_t

No model is retrained online. If no candidate passes the current filters, GDC uses the known-feasible fallback. The optional support-aware variant may also downweight learned guidance when its empirical support screen rejects the pool.

The offline stage has one purpose: turn long feasible behavior into a compact, anchored distribution of local progress patterns. Rather than assuming complete physics or globally solved optimal trajectories, GDC starts from known-feasible trajectories and cuts them into short snippets. The known constraints define the feasible scaffold on which those snippets can live. The progress certificates then tilt a probabilistic path model toward snippets that move in locally preferable directions. The probabilistic path approximates the missing or unobserved physics statistically; the known physics remains explicit as constraints that reduce and anchor the search space.

Thus the offline stage trains the objects that will later be frozen and used by the online controller. It has two logically separate components:

1. **Feasible action-segment embedding:** an encoder-decoder learns a compact latent coordinate system $\widehat{\mathcal{M}}^*$ for known-feasible action segments.
2. **Progress density learning:** a progress-tilted probabilistic path learns how short feasible snippets tend to move on $\widehat{\mathcal{M}}^*$ and supplies the score used online.

Diffusion is therefore not the method itself and not a second encoder path. It is one possible gradual implementation of the probabilistic path.

Progress-tilted target distribution. For contexts with $H^\star(o, g, \mathcal{C}_{\text{known}}) \neq \varnothing$, let $q_{\mathcal{D}}^{H^\star}(\tau \mid o, g, \mathcal{C}_{\text{known}})$ denote the empirical distribution of known-feasible snippets at the selected horizon. A signed progress certificate $r = \Delta\rho(\tau, g)$ defines the progress-tilted distribution

$$q_\beta(\tau \mid o, g, \mathcal{C}_{\text{known}}, H^\star) = \frac{q_{\mathcal{D}}^{H^\star}(\tau \mid o, g, \mathcal{C}_{\text{known}})}{Z_\beta(o, g, \mathcal{C}_{\text{known}}, H^\star)} \times \exp(\beta \,\text{clip}(\Delta\rho(\tau, g), -c, c)) \,, \tag{30}$$

restricted to snippets satisfying the known constraints. Thus the learned path is constraint-aware because its training support and conditioning variables include $\mathcal{C}_{\text{known}}$, but it is not itself a certificate of hard feasibility outside the empirical support. This is a soft tilt, not a hard filter. Positive-progress snippets receive larger likelihood, but temporarily regressive snippets may remain when they represent waiting, yielding, backing up, or reorientation. The target is not "every snippet improves." The target is that the learned conditional distribution has positive expected progress in the data-supported region under the stated positive-progress assumption. With $\tilde{r}(\tau, g) = \text{clip}(\Delta\rho(\tau, g), -c, c)$,

$$\mathbb{E}_{q_\beta}[\tilde{r}] = \frac{\partial}{\partial\beta} \log Z_\beta, \qquad \frac{\partial}{\partial\beta}\mathbb{E}_{q_\beta}[\tilde{r}] = \text{Var}_{q_\beta}[\tilde{r}] \geq 0. \tag{31}$$

Thus the assumption (22) is invoked only for a selected, supported context: the tilted distribution already has positive expected progress under that assumption, and the exponential tilt amplifies the weight on larger certificates without deleting necessary non-progressing maneuvers. Equivalently, for $\beta \geq 0$, the training target places more probability mass on larger progress certificates while remaining restricted to $\mathcal{C}_{\text{known}}$.

Feasible action-segment embedding. The latent action-segment representation is essential for scalability. Directly modeling or searching over a horizon-H action segment lives in a space of dimension Hd_u, which can be prohibitive for multilevel optimization or autonomous driving. GDC instead learns a low-dimensional latent variable $z \in \mathcal{Z}$ with $d_z \ll Hd_u$, and trains the probabilistic path on the latent distribution induced by feasible-progress snippets through this encoder. The online controller therefore samples and refines z, then decodes z into an executable action segment.

When the known physics has geometric coordinates, this latent space can be realized as

$$\mathcal{Z} = \mathbb{R}^{d_0} \times G_1 \times \cdots \times G_L, \tag{32}$$

where the Euclidean factors capture unconstrained latent variation and the Lie group factors capture known geometric degrees of freedom such as pose, phase, orientation, or network-flow coordinates. This product/geometric structure is optional; the compact latent action-segment map is not. The essential scaffold is the known-feasible set $\widehat{\mathcal{M}}^*_{\text{known}}(o, g)$ inside this learned low-dimensional search space.

GDC does not need a distillation loss from a complete solution map. The offline object being embedded is already the behavior segment that will be controlled online: a known-feasible short action trajectory. Since no ground-truth latent code is observed, we learn the latent scaffold with a regularized variational action-segment autoencoder, where H_k is the selected horizon of the training segment τ_k:

$$q_\xi(z \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}}), \qquad p_\psi(\tau_k \mid z, o_k, g_k, \mathcal{C}_{\text{known}}, H_k). \tag{33}$$

For notation, a latent code can be sampled from the encoder or set to its posterior mean:

$$z_k \sim q_\xi(\cdot \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}}), \qquad \bar{z}_k = \mathbb{E}_{q_\xi}[z \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}}]. \tag{34}$$

When deterministic notation is convenient, we write $E_\xi(\tau_k, o_k, g_k, \mathcal{C}_{\text{known}}) = \bar{z}_k$ only as shorthand for this posterior mean; the learned encoder itself is the variational distribution q_ξ. The decoder distribution reconstructs an executable short action segment:

$$\hat{\tau}_k = D_\psi(z_k, o_k, g_k, \mathcal{C}_{\text{known}}, H_k) \quad \text{or equivalently} \quad \hat{\tau}_k \sim p_\psi(\cdot \mid z_k, o_k, g_k, \mathcal{C}_{\text{known}}, H_k), \tag{35}$$

The first action of the decoded segment is the executable control. State reconstruction is not a primary objective; states enter only through the observation, the known constraints, and the progress certificate used to label the segment.

Structured offline objective. The feasible map is trained with only the losses needed to make the action-segment coordinates smooth, executable, and known-feasible:

$$\mathcal{L}_{\text{map}} = \mathcal{L}_{\text{vae}} + \lambda_k \mathcal{L}_{\text{known}} \tag{36}$$

Here

$$\begin{aligned} \mathcal{L}_{\text{vae}} = \mathbb{E}_k \big[& \mathbb{E}_{z \sim q_\xi(\cdot \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}})} \left[-\log p_\psi(\tau_k \mid z, o_k, g_k, \mathcal{C}_{\text{known}}, H_k) \right] \\ & + \beta_z \, \mathrm{KL}(q_\xi(z \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}}) \,\|\, p_0(z)) \big], \end{aligned} \tag{37}$$

where $p_0(z)$ is a simple prior such as $\mathcal{N}(0, I)$, or a structured prior when the latent coordinates include known geometric factors. The known-constraint loss penalizes decoded action segments that violate $\mathcal{C}_{\text{known}}$:

$$\mathcal{L}_{\text{known}} = \mathbb{E}_k \mathbb{E}_{z \sim q_\xi(\cdot \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}})} \left[\sum_r \left[f_r^{\text{known}}(D_\psi(z, o_k, g_k, \mathcal{C}_{\text{known}}, H_k), o_k, g_k)_+ \right]^2 \right]. \tag{38}$$

The KL term and the low-dimensional bottleneck prevent the encoder from becoming an identity map and make the latent space suitable for sampling. Additional smoothness or contrastive regularizers may be added, but they are not progress losses.

After minimizing (36), the empirical manifold $\widehat{\mathcal{M}}^*$ is the data-supported region of the latent space reached by known-feasible action segments.

We separate $\mathcal{L}_{\text{map}}$ from the path objective because they answer different questions. The map loss learns only the coordinate system for known-feasible action segments. It does not upweight positive progress or enforce positive expected progress. The path loss then learns a conditional density on that scaffold: where progress-biased feasible behavior has probability mass, and therefore which direction the controller should move online. Conceptually GDC first learns the feasible action-segment map and then learns the progress-tilted probability path over that map.

Progress-tilted probabilistic path. After the map exists, GDC fits a progress-tilted probabilistic path over the latent codes on $\widehat{\mathcal{M}}^*$. This path learns the recurring local movement patterns inside the segmented feasible trajectories: how one known-feasible local behavior tends to transition toward another under the same goal and constraint context. These patterns are the statistical approximation of the missing physics and cross-level couplings that were not included in $\mathcal{C}_{\text{known}}$. The known physics does not merely appear as a conditioning variable; it shrinks and anchors the path. Define the known-feasible latent scaffold

$$\widehat{\mathcal{M}}^*_{\text{known}}(o,g) = \left\{ z \in \widehat{\mathcal{M}}^* : f_r^{\text{known}}(D_\psi(z,o,g,\mathcal{C}_{\text{known}},H^\star),o,g) \le 0,\ r=1,\dots,R \right\}. \quad (39)$$

The probability path is learned on, and sampled back toward, this anchored scaffold:

$$p_\theta^Z(z \mid o,g,\mathcal{C}_{\text{known}},H^\star) \text{ is supported or strongly concentrated on } \widehat{\mathcal{M}}^*_{\text{known}}(o,g). \quad (40)$$

Thus the path is not asked to rediscover the known physics from data or explore the full trajectory space. The known constraints act as fixed anchors that hold the distribution to the feasible scaffold, while the progress tilt learns how feasible short segments move across the parts of the system whose physics is missing or only partially observed.

A diffusion-style path is one implementation: it gradually approximates the clean feasible-progress distribution by moving along a sequence of corrupted distributions. For a path step ℓ and a clean latent point $z_k \sim q_\xi(\cdot \mid \tau_k, o_k, g_k, \mathcal{C}_{\text{known}})$,

$$z_{k,\ell} = \sqrt{\bar\alpha_\ell}\, z_k + \sqrt{1-\bar\alpha_\ell}\,\epsilon, \qquad \epsilon \sim \mathcal{N}(0,I). \quad (41)$$

Following recent direct clean-data prediction views of generative path models, the network predicts the clean latent behavior rather than the injected perturbation. Let

$$\begin{aligned} \hat z_{0,k} = F_\theta(z_{k,\ell},o_k,g_k,\mathcal{C}_{\text{known}},H_k,\ell), \qquad & \hat\tau_{0,k} = D_\psi(\hat z_{0,k},o_k,g_k,\mathcal{C}_{\text{known}},H_k), \\ & \hat u_{0,k} = [\hat\tau_{0,k}]_0. \end{aligned} \quad (42)$$

The progress-weighted clean-behavior prediction objective is anchored by the known constraints:

$$\mathcal{L}_{\text{path}} = \mathbb{E}_{k,\ell,\epsilon}[\tilde w_k\, \mathcal{B}_k], \qquad \tilde w_k = \frac{\exp(\beta\, \text{clip}(r_k,-c,c))}{\mathbb{E}_j[\exp(\beta\, \text{clip}(r_j,-c,c))]}. \quad (43)$$

where

$$\begin{aligned} \mathcal{B}_k = \|\hat z_{0,k} - z_k\|^2 + \lambda_\tau \|D_\psi(\hat z_{0,k},o_k,g_k,\mathcal{C}_{\text{known}},H_k) - \tau_k\|^2 \\ + \lambda_{\text{anc}} \mathcal{A}_{\text{known}}(\hat z_{0,k},\hat u_{0,k},\hat\tau_{0,k}). \end{aligned} \quad (44)$$

The anchor penalty is

$$\mathcal{A}_{\text{known}}(\hat z,\hat u,\hat\tau) = \sum_r \left[f_r^{\text{known}}(\hat u,o,g)_+\right]^2 + \sum_t \sum_r \left[f_r^{\text{known}}(\hat\tau_t,o,g)_+\right]^2. \quad (45)$$

The weighted objective is the empirical implementation of the tilted target (30). Let q_β^Z denote the progress-weighted aggregate posterior induced by the variational encoder:

$$q_\beta^Z(z \mid o,g,\mathcal{C}_{\text{known}},H^\star) = \int q_\xi(z \mid \tau,o,g,\mathcal{C}_{\text{known}})\, q_\beta(\tau \mid o,g,\mathcal{C}_{\text{known}},H^\star)\, \mathrm{d}\tau. \quad (46)$$

Minimizing (43) fits the latent path density $p_\theta^Z(z \mid o,g,\mathcal{C}_{\text{known}},H^\star)$ to this tilted latent distribution, subject to the known anchors (39)–(45). Let p_θ^τ denote the decoded segment distribution induced by $z \sim p_\theta^Z(\cdot \mid o,g,\mathcal{C}_{\text{known}},H^\star)$ and $\tau = D_\psi(z,o,g,\mathcal{C}_{\text{known}},H^\star)$. If the decoded model is sufficiently close to the tilted distribution in this context, the assumed data moment (22) yields the following progress-bias property:

$$\mathbb{E}_{\tau\sim p_\theta^\tau(\cdot\mid o,g,\mathcal{C}_{\text{known}},H^\star)}[\Delta\rho(\tau,g)] \approx \mathbb{E}_{\tau\sim q_\beta(\cdot\mid o,g,\mathcal{C}_{\text{known}},H^\star)}[\Delta\rho(\tau,g)] > 0 \quad (47)$$

inside the data-supported known-feasible region. The approximation error is the density fitting error of the probabilistic path model plus decoder error from the latent map. This objective gives larger density to latent regions associated with positive signed progress while retaining feasible waiting, yielding, or repositioning snippets when they are present in the data. The result is a conditional latent density $p_\theta^Z(z \mid o,g,\mathcal{C}_{\text{known}},H^\star)$ and score $s_\theta(z,o,g,\mathcal{C}_{\text{known}},H^\star) = \nabla_z \log p_\theta^Z(z \mid o,g,\mathcal{C}_{\text{known}},H^\star)$ on the learned map. This score is the online progress direction used by GDC.

Offline output. The offline stage returns the action-segment posterior model $q_\xi(z \mid \tau, o, g, \mathcal{C}_{\text{known}})$, the decoder D_ψ (or likelihood p_ψ), the clean latent predictor F_θ, and the conditional path score $s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star)$. The empirical manifold $\widehat{\mathcal{M}}^*$ is the data-supported, known-feasible region of the action-segment map, equipped with a progress-tilted path density.

C.2 Geometry and progress score

Definition C.1 (Pullback metric on $\widehat{\mathcal{M}}^*$)**.** *The Riemannian metric on $\widehat{\mathcal{M}}^*$ is the decoder-pullback metric:*

$$G(z) = J_D(z)^\top J_D(z) + \lambda_m I, \tag{48}$$

where J_D is the Jacobian of the action-segment decoder, for example $D(z) = D_\psi(z, o, g, \mathcal{C}_{\text{known}}, H^\star)$, and $\lambda_m > 0$ ensures positive definiteness.

For a smooth function $\Phi : \widehat{\mathcal{M}}^* \to \mathbb{R}$, the Riemannian gradient is

$$\operatorname{grad}_G \Phi(z) = G(z)^{-1} \nabla_z \Phi(z), \tag{49}$$

and the Riemannian gradient flow is

$$\dot{z} = -\operatorname{grad}_G \Phi(z) = -G(z)^{-1} \nabla_z \Phi(z). \tag{50}$$

Proposition C.2 (Invariance)**.** *The Riemannian gradient flow* (50) *is invariant under diffeomorphic reparameterizations of $\widehat{\mathcal{M}}^*$.*

Proof. Let $\varphi : \widetilde{\mathcal{M}} \to \widehat{\mathcal{M}}^*$ be a C^1 diffeomorphism and write $z = \varphi(\tilde{z})$. The pullback metric is $\tilde{G}(\tilde{z}) = J_\varphi(\tilde{z})^\top G(z) J_\varphi(\tilde{z})$ and the pulled-back potential is $\tilde{\Phi}(\tilde{z}) = \Phi(\varphi(\tilde{z}))$. By the chain rule, $\nabla_{\tilde{z}} \tilde{\Phi} = J_\varphi^\top \nabla_z \Phi$. The gradient vector field in the $\tilde{z}$ coordinates is the unique vector $\widetilde{X}$ satisfying $\tilde{G}(\widetilde{X}, \widetilde{v}) = \mathrm{d}\tilde{\Phi}[\widetilde{v}]$ for every tangent vector $\widetilde{v}$. Substituting the definitions gives $J_\varphi \widetilde{X} = G^{-1} \nabla_z \Phi = \operatorname{grad}_G \Phi$. Therefore the pushed-forward flow obeys $\frac{\mathrm{d}}{\mathrm{d}t} \varphi(\tilde{z}_t) = J_\varphi \dot{\tilde{z}}_t = -\operatorname{grad}_G \Phi(z_t)$, which is exactly (50) in the original coordinates. □

Offline progress-path implementation and score extraction. This subsection specifies one offline implementation of the progress-tilted latent path. After training, the online controller treats q_ξ, D_ψ, $H^\star$, and p_θ^Z as fixed. The learned object is the progress-tilted latent path density $p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$; the online score $s_\theta = \nabla_z \log p_\theta^Z$ is derived from this density. Known physics is not left implicit in the data distribution: it appears as constraints, barriers, anchors, and filters. The path density approximates what is missing from the explicit model: implicit physics, latent preference structure, cross-level coupling, and long-horizon directional information.

Diffusion-style gradual path instantiation. Let $z_0 \sim q_{\text{data}}(z)$ denote a latent behavior sampled from the data distribution on $\widehat{\mathcal{M}}^*$. A convenient implementation is a diffusion-style gradual path: one moves from clean feasible-progress behaviors to corrupted latent points and learns to predict the clean behavior back. This is not required by GDC; any differentiable conditional density or path model that yields $\nabla_z \log p_\theta^Z$ can be used. For the diffusion-style implementation, the forward path adds noise within the tangent bundle $T\widehat{\mathcal{M}}^*$:

$$q(z_\ell \mid z_{\ell-1}) = \mathcal{N}_{\widehat{\mathcal{M}}^*}\Big(z_\ell;\ \sqrt{1-\beta_\ell}\, z_{\ell-1},\ \beta_\ell G(z_{\ell-1})^{-1}\Big), \tag{51}$$

where $\mathcal{N}_{\widehat{\mathcal{M}}^*}$ denotes the Riemannian normal distribution on $\widehat{\mathcal{M}}^*$, and $G(z)^{-1}$ ensures noise is scaled according to the local geometry.

In practice, when $\widehat{\mathcal{M}}^*$ is approximately flat at the scale of the noise (which holds for well-trained manifolds with sufficient latent dimension), we use the Euclidean approximation:

$$q(z_\ell \mid z_{\ell-1}) \approx \mathcal{N}\Big(z_\ell;\ \sqrt{1-\beta_\ell}\, z_{\ell-1},\ \beta_\ell I\Big), \tag{52}$$

with $\{\beta_\ell\}_{\ell=1}^T$ a noise schedule, $\alpha_\ell = 1 - \beta_\ell$, and $\bar{\alpha}_\ell = \prod_{j=1}^{\ell} \alpha_j$. The marginal is

$$q(z_\ell \mid z_0) = \mathcal{N}\big(z_\ell;\ \sqrt{\bar{\alpha}_\ell}\, z_0,\ (1-\bar{\alpha}_\ell) I\big). \tag{53}$$

Clean latent predictor fixed after offline training. The probabilistic path is a model of *what action or short action trajectory should be executed next*, conditioned on the current observation, goal, and known constraints. It is not primarily a next-state predictor, and it does not directly predict the full high-dimensional action segment as its core output. The path network predicts the clean latent behavior, and the fixed decoder D_ψ maps that latent point to the executable action segment. This choice follows the clean-data prediction principle advocated by Li & He (2026): clean data lie on the low-dimensional data manifold, while corrupted quantities generally do not. In GDC, this means predicting clean latent action-segment codes, not merely a latent perturbation residual and not a full action sequence in ambient dimension:

$$\hat{z}_0 = F_\theta(z_\ell, o, g, \mathcal{C}_{\text{known}}, H^\star, \ell), \qquad \hat{\tau}_0 = D_\psi(\hat{z}_0, o, g, \mathcal{C}_{\text{known}}, H^\star), \qquad \hat{u}_0 = [\hat{\tau}_0]_0. \tag{54}$$

Offline training uses this clean latent target to implement the path objective (43): for a corrupted latent point $z_\ell = \sqrt{\bar{\alpha}_\ell} z_0 + \sqrt{1-\bar{\alpha}_\ell}\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, the model predicts z_0 and weights the latent prediction by the progress tilt induced by the signed certificate. A decoded trajectory loss through D_ψ may be used as a consistency term, but direct action prediction is not the core path model. State quantities are not part of the core path loss; when available, they are used only to evaluate known dynamics, check constraints, or compute progress certificates. During online control F_θ and D_ψ are fixed; the clean predictor may be queried to estimate the path score, but it does not replace the online latent control dynamics.

Remark C.3 (Known constraints versus learned feasibility)**.** *Since every training sample z_0 lies on $\widehat{\mathcal{M}}^*$ and is filtered or penalized by $\mathcal{C}_{\text{known}}$, the learned path concentrates its density on the data-supported progress-biased region. This does not prove feasibility for all implicit physics; rather, it makes implicit feasibility a statistical property of feasible trajectory demonstrations with progress certificates. Known constraints are still enforced explicitly through conditioning, barrier terms, anchor penalties, and candidate filtering during control.*

Score induced by clean behavior prediction. Although the network predicts clean behavior directly, it still induces a score field for online control. For the diffusion-style gradual path, under the Gaussian forward kernel $z_\ell = \sqrt{\bar{\alpha}_\ell} z_0 + \sqrt{1-\bar{\alpha}_\ell}\epsilon$, replacing z_0 by its clean prediction gives the score approximation

$$s^{\text{diff}}_{\theta,\ell}(z_\ell, o, g, \mathcal{C}_{\text{known}}, H^\star) \approx \frac{\sqrt{\bar{\alpha}_\ell}\, F_\theta(z_\ell, o, g, \mathcal{C}_{\text{known}}, H^\star, \ell) - z_\ell}{1-\bar{\alpha}_\ell}. \tag{55}$$

Thus GDC can use direct action/trajectory prediction during training while still obtaining the score direction needed for Riemannian gradient-flow control.

C.3 Online progress-energy navigation

At deployment, all learned objects are frozen. GDC receives the action-segment posterior q_ξ, decoder D_ψ, candidate horizon pool $\mathcal{H}$, progress-tilted latent density p^Z_θ, and score $s_\theta = \nabla_z \log p^Z_\theta$. The online stage is therefore not another learning or diffusion-sampling procedure. It is a geometric navigator on the learned feasible-progress map: the controller localizes the current context, forms a composable latent energy, follows its induced vector field for a fixed number of small steps, decodes an action segment, and applies the final known-constraint projection before executing the first action.

Context-consistent latent localization. At time t, the controller observes o_t and constructs a local data-supported candidate set for each candidate horizon $H \in \mathcal{H}$. For this candidate-specific construction, write $H^\star_t = H$:

$$\mathcal{M}_t = \left\{ z \in \widehat{\mathcal{M}}^* : D_\psi(z, o_t, g, \mathcal{C}_{\text{known}}, H^\star_t) \text{ is locally known-feasible} \right\}. \tag{56}$$

In implementation, $\mathcal{M}_t$ is represented by warm-starts from the previous accepted latent state, samples from $p^Z_\theta(\cdot \mid o_t, g, \mathcal{C}_{\text{known}}, H^\star_t)$, or nearby latent codes whose decoded segments pass the known-constraint filter. This plays the role of manifold localization: it anchors online navigation to the part of the learned map that is consistent with the current observation and known constraints.

Online energy and nominal vector field. The direction of online motion is determined by the composite latent energy

$$\begin{aligned}\Phi_{\mathrm{on},t}(z) = \frac{1}{\epsilon_{\mathrm{s}}}\Phi_{\mathrm{scaf}}(z; o_t, \mathcal{C}_{\mathrm{known}}) + \lambda\Phi_{\mathrm{known}}(z; o_t, \mathcal{C}_{\mathrm{known}}) \\ + (1-\lambda)\Phi_{\mathrm{data}}(z; o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) + \Phi_{\mathrm{task}}(z; o_t, g),\end{aligned} \tag{57}$$

where Φ_{scaf} is a fast restoration term that keeps z near $\mathcal{M}_t$, Φ_{known} contains available physics and explicit constraints, Φ_{task} encodes the runtime objective, and

$$\Phi_{\mathrm{data}}(z; o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) = -\log p_\theta^Z(z \mid o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) \tag{58}$$

is the progress energy learned offline. The nominal vector field is the Riemannian descent flow

$$\begin{aligned}F_t(z) := \dot{z} &= -\operatorname{grad}_G \Phi_{\mathrm{on},t}(z) \\ &= -G(z)^{-1}\begin{bmatrix}\frac{1}{\epsilon_{\mathrm{s}}}\nabla_z\Phi_{\mathrm{scaf}}(z) + \lambda\nabla_z\Phi_{\mathrm{known}}(z) \\ \quad -(1-\lambda)s_\theta(z, o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) + \nabla_z\Phi_{\mathrm{task}}(z)\end{bmatrix}.\end{aligned} \tag{59}$$

The scalar ϵ_{s} induces the same snap-then-slide structure as a geometric navigator: the scaffold term rapidly restores the latent state to the known-feasible support, while the data and task terms move tangentially toward progress. The learned path can therefore be kept fixed while new physics, safety, preference, or resource terms are added to the energy.

Geometry-aware latent evolution. Given F_t, GDC advances each candidate latent state with a fixed-depth integrator rather than an online nonlinear optimizer. A simple second-order implementation is

$$k_1 = \Pi_{\|\cdot\|_G \le V_{\max}} F_t(z_n), \tag{60}$$

$$z_{\mathrm{mid}} = z_n + \tfrac{\Delta t}{2}\, k_1, \tag{61}$$

$$k_2 = \Pi_{\|\cdot\|_G \le V_{\max}} F_t(z_{\mathrm{mid}}), \tag{62}$$

$$\Delta z_n = \Delta t\, k_2, \tag{63}$$

where $\Pi_{\|\cdot\|_G \le V_{\max}}$ caps the latent velocity in the decoder-pullback metric. The increment is then integrated in the latent chart or product geometry:

$$\bar{z}_{n+1} = z_n \oplus \Delta z_n, \tag{64}$$

where $\oplus$ is ordinary addition in a flat latent chart and the corresponding group or retraction update when structured latent factors are used. Finally, the integrated point is accepted only after a scaffold correction:

$$z_{n+1} = (1-\mu_R)\bar{z}_{n+1} + \mu_R\, \Pi_{\mathcal{M}_t}(\bar{z}_{n+1}), \tag{65}$$

with $\mu_R \in [0,1]$. In the common case where $\bar{z}_{n+1}$ already decodes to a known-feasible segment, $\Pi_{\mathcal{M}_t}$ is the identity. Otherwise it is implemented by the simplest problem-specific retraction, clipping, or backtracking rule. Thus the bridge from the differential law to decoding is explicit:

$$\dot{z} \;\longrightarrow\; \Delta z_n \;\longrightarrow\; \bar{z}_{n+1} \;\longrightarrow\; z_{n+1} \in \mathcal{M}_t \;\longrightarrow\; D_\psi(z_{n+1}, o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star).$$

The velocity cap and retraction are not optimizers; they are numerical guards that keep the online state in the region where the decoder and score are trusted.

Decode, project, and execute. After N_r latent evolution steps, each candidate has an accepted post-navigation latent state $z_{t,\mathrm{nav}}^{(k)} := z_{N_r}^{(k)} \in \mathcal{M}_t$. This is the latent point that is decoded:

$$\tau_t^{(k)} = D_\psi(z_{t,\mathrm{nav}}^{(k)}, o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star). \tag{66}$$

GDC selects a feasible-progress candidate using the task score J_{sel} and takes the first decoded action $u_t^{\mathrm{nom}} = \tau_{t,0}^{\mathrm{nom}}$. The final executed action is the weighted projection onto the current known-feasible action set:

$$u_t = \Pi^W_{\mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})}(u_t^{\mathrm{nom}}) := \operatorname*{arg\,min}_{u \in \mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})} \frac{1}{2}\|u - u_t^{\mathrm{nom}}\|_W^2. \tag{67}$$

Algorithm 2 GDC online progress-energy navigation

Input: current observation o_t, goal g, previous action u_{t-1}, frozen models $(q_\xi, D_\psi, p_\theta^Z, s_\theta)$, candidate horizons, known-constraint checker, projection layer, and selection cost J_{sel}.
Output: executable action u_t.
Form the available horizon pool $\mathcal{H}_t$, optionally screened by the support test. If $\mathcal{H}_t = \varnothing$, return the known-feasible fallback.
Build or warm-start K candidates $z_0^{(1:K)}$ across $H \in \mathcal{H}_t$ in their local known-feasible sets.
Form each candidate's $\Phi_{\mathrm{on},t}$ and Riemannian vector field $F_t = -\operatorname{grad}_G \Phi_{\mathrm{on},t}$.
for $k = 1, \dots, K$ **do**
    Take N_r capped RK2/retraction steps using Eqs. (60)–(65).
    Decode $\tau_t^{(k)} = D_\psi(z_{N_r}^{(k)}, o_t, g, \mathcal{C}_{\mathrm{known}}, H_k)$ and discard candidates that violate $\mathcal{C}_{\mathrm{known}}$.
end for
If no decoded candidate is feasible, return the known-feasible fallback action.
Select a feasible segment minimizing $J_{\mathrm{sel}}(\tau, o_t, g)$ across horizons and set u_t^{nom} to its first action.
Project $u_t = \Pi^W_{\mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})}(u_t^{\mathrm{nom}})$; if projection fails, return the known-feasible fallback action.
Execute u_t and replan at the next observation.

For box constraints this is clipping; for equality constraints it has the standard weighted-projection closed form. If no decoded candidate or final projection is feasible, the controller resamples or uses the known-feasible backup. The projection is only the last execution guard: the main online controller is the energy-induced latent navigation in (59).

In the reported SUMO multi-H implementation, candidate generation and selection range over $H \in \{6, 12, 24\}$, with a runtime candidate-progress gate.

The dominant online cost is fixed-depth navigation: $O(KN_r(C_\theta + C_{\mathrm{known}}) + C_{\mathrm{proj}})$. No model is retrained online and no full nonlinear planning problem is solved. The repeated online operation is evaluating the frozen score/energy, integrating a bounded latent vector field, decoding candidate action segments, and applying the final known-constraint projection.

Remark C.4 (Transfer by replacing the known scaffold)**.** *The separation between direction and feasibility gives a precise transfer interface. Let $\mathcal{C}^{\mathrm{tr}}_{\mathrm{known}}$ be the training scaffold and $\mathcal{C}^{\mathrm{dep}}_{\mathrm{known}}$ the current deployment scaffold. Transfer freezes the learned parameters and recomputes every explicit object from the deployment description:*

$$\begin{aligned} &\hat\theta, \hat\xi, \hat\psi \text{ fixed}, && c^{\mathrm{dep}}_{t,H} = (o_t, g, \mathcal{C}^{\mathrm{dep}}_{\mathrm{known}}, H), \quad H \in \mathcal{H}, \\ &z^{\mathrm{dep}}_{n+1} = \mathrm{RK2Retract}(z_n, F_t(\cdot; c^{\mathrm{dep}}_{t,H}), \mathcal{M}^{\mathrm{dep}}_{t,H}), && u_t = \mathrm{Guard}_{\mathcal{C}^{\mathrm{dep}}_{\mathrm{known}}}(D_{\hat\psi}(z^{\mathrm{dep}}_{N_r}, c^{\mathrm{dep}}_{t,H})). \end{aligned} \tag{68}$$

Thus a changed map, constraint set, or analytic response model alters candidate ranking, barriers, supported candidates, checks, projection, and fallback. The learned score is queried conditionally on that new description, but it is never allowed to override it. The reported transfer evaluations supply the current map and disturbances as $\mathcal{C}^{\mathrm{dep}}_{\mathrm{known}}$ and reuse the frozen score. Their observed performance supports transfer on those tested contexts; the analysis requires score coverage in the new context. The optional support screen can abstain in a poorly supported neighborhood.

D Proofs of the Main-Text Statements

This section states the conditions used in Section 4 and proves the results stated there. These results are deliberately elementary, and we do not present them as technical contributions: each follows in a few lines from an explicitly stated assumption. Theorem D.3 is an identity under the modeling assumption $p^\star \propto Z_H$ followed by the Cauchy–Schwarz inequality; Proposition D.4 holds by construction of the final projection and the known-feasible fallback; Proposition D.5 restates the acceptance rule of the retraction/backtracking guard; and Theorem D.6 is a standard strong-convexity argument. Their role is bookkeeping: they separate the properties that the controller enforces by construction (known-feasible execution and bounded scaffold deviation) from those that rest on modeling and approximation assumptions (the direction supplied by the learned score and closeness to an oracle optimum). Expanded derivations and additional results are given in Appendix E.

Assumption D.1 (Data-supported feasible-progress scaffold)**.** *For each context $(o, g, \mathcal{C}_{\mathrm{known}})$ in the modeled operating envelope, the offline dataset contains known-feasible action segments whose*

decoded support lies within the explicit known-constraint scaffold. For a selected candidate horizon $H^\star$, assume that the progress-tilted conditional distribution has positive expected signed progress:

$$\mathbb{E}_{\tau\sim q_\beta(\cdot|o,g,\mathcal{C}_{\text{known}},H^\star)}[\Delta\rho(\tau,g)] > 0. \tag{69}$$

This assumption is required for the directional claim; the empirical support screen above does not establish it as a statistical guarantee.

Assumption D.2 (Regular latent map and explicit execution guard)**.** *The decoder D_ψ is Lipschitz on the data-supported neighborhood of $\widehat{\mathcal{M}}^*$, the known-constraint functions are locally Lipschitz after decoding, and the online controller uses bounded latent velocity, scaffold retraction or backtracking, decoded candidate filtering, and a final projection onto the current known-feasible action set $\mathcal{U}_{\text{known}}(o_t, u_{t-1})$. This set is closed and nonempty whenever the projection layer is invoked, and the fallback action belongs to it.*

Theorem D.3 (Progress-measure score and local descent)**.** *Under Assumption D.1 and a maximum-entropy model of feasible progress demonstrations, suppose the continuation partition function is finite, positive, and C^1 in z on the selected data-supported neighborhood, and that it induces $p^\star(z \mid c) \propto Z_H(z,c)$ with respect to the chosen latent reference measure for the selected context $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$. If the learned latent path satisfies $\|\nabla_z \log p_\theta^Z - \nabla_z \log p^\star\| \le \varepsilon_{\text{approx}}$ on the data-supported neighborhood, then the progress-tilted path density defines a data-induced soft progress value $V_{\text{prog}}^{\text{soft}}$ on $\widehat{\mathcal{M}}^*$ such that*

$$s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) = \nabla_z \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star) = -\nabla_z V_{\text{prog}}^{\text{soft}}(z,c) + \mathcal{O}(\varepsilon_{\text{approx}}), \tag{70}$$

where $\varepsilon_{\text{approx}}$ is the path-model approximation error. Moreover, let $e_\theta = s_\theta + \nabla_z V_{\text{prog}}^{\text{soft}}$ and suppose $\|e_\theta\|_{G^{-1}} \le \varepsilon_{\text{approx}}$. Along the score-only natural-gradient component $\dot z = G^{-1} s_\theta$,

$$\frac{\mathrm{d}}{\mathrm{d}t} V_{\text{prog}}^{\text{soft}}(z_t, c) \le -a_t(a_t - \varepsilon_{\text{approx}}), \qquad a_t = \|\operatorname{grad}_G V_{\text{prog}}^{\text{soft}}(z_t, c)\|_G. \tag{71}$$

Thus the learned component is a strict descent direction outside the approximation-error tube $a_t \le \varepsilon_{\text{approx}}$.

Proof. Fix a supported context $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$ at a selected candidate horizon. Under the maximum-entropy demonstration model, feasible future segments starting from latent point z have an unnormalized partition function $Z_H(z,c) = \int \exp[-C_H(\tau; c)]\, \mathrm{d}\nu_z(\tau)$ over known-feasible continuations, with soft progress value $V_{\text{prog}}^{\text{soft}}(z,c) = -\log Z_H(z,c)$. The model posits that the induced ideal latent density is proportional to this partition function: $p^\star(z \mid c) = Z_c^{-1} \exp[-V_{\text{prog}}^{\text{soft}}(z,c)]$. Hence $\nabla_z \log p^\star(z \mid c) = -\nabla_z V_{\text{prog}}^{\text{soft}}(z,c)$. If the learned path satisfies $\|\nabla_z \log p_\theta^Z - \nabla_z \log p^\star\| \le \varepsilon_{\text{approx}}$ on the data-supported neighborhood, then (70) follows. For the second claim,

$$\frac{\mathrm{d}}{\mathrm{d}t} V_{\text{prog}}^{\text{soft}} = \left\langle \nabla V_{\text{prog}}^{\text{soft}}, G^{-1}(-\nabla V_{\text{prog}}^{\text{soft}} + e_\theta) \right\rangle \tag{72}$$

$$\le -\|\operatorname{grad}_G V_{\text{prog}}^{\text{soft}}\|_G^2 + \|\operatorname{grad}_G V_{\text{prog}}^{\text{soft}}\|_G \|e_\theta\|_{G^{-1}}, \tag{73}$$

which gives (71). Appendix E.2 gives the full regularized statement. □

Theorem D.3 adds no content beyond its modeling assumption. It records that, when the progress-tilted target is identified with $p^\star$ (an assumption, not a consequence of training), the learned score is an approximate descent direction for the soft progress value of the data policy. This is the sense in which the learned distribution can choose among locally known-feasible options whose explicit constraints alone do not determine a direction.

Proposition D.4 (Known-feasible online execution)**.** *Under Assumption D.2, every action executed by Algorithm 2 belongs to the current known-feasible action set whenever the projection layer is available:*

$$u_t \in \mathcal{U}_{\text{known}}(o_t, u_{t-1}). \tag{74}$$

If no decoded candidate or projected action is feasible, the controller resamples or uses the known-feasible fallback.

Proof. Algorithm 2 decodes only candidate segments that pass the known-constraint filter. Its final executed command is $u_t = \Pi^W_{\mathcal{U}_{\text{known}}(o_t,u_{t-1})}(u_t^{\text{nom}})$ whenever the projection exists. By definition of projection onto a nonempty closed known-feasible set, the projected point belongs to $\mathcal{U}_{\text{known}}(o_t, u_{t-1})$. If the projection layer is unavailable or fails, or if no candidate survives filtering, the algorithm returns the known-feasible fallback required by Assumption D.2. Therefore every executed non-fallback or fallback action lies in the current known-feasible action set. □

Proposition D.5 (Per-step scaffold deviation)**.** *Under the retraction/backtracking condition of Proposition E.21, each accepted latent point satisfies*

$$d_G(z_t, \mathcal{M}_t) \le \sqrt{\epsilon_s/2}\, V_{\max} \Delta t. \tag{75}$$

The bound is on distance to the known-feasible latent scaffold. A bound relative to an ideal action additionally needs a specified target and a tracking condition for the local refinement.

Theorem D.6 (Approximate optimality)**.** *Fix a selected context $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$ and a compact geodesically convex data-supported region $U_c \subset \widehat{\mathcal{M}}^*$. Let $\Phi_{\text{opt}}(\cdot, c)$ denote the oracle constrained horizon objective on U_c, and let $z^\star = \arg\min_{z\in U_c} \Phi_{\text{opt}}(z, c)$ be its unique minimizer. Suppose the learned GDC potential approximates the oracle objective up to the context-dependent normalizing constant $a(c)$:*

$$\sup_{z\in U_c} |\Phi_{\text{GDC}}(z, c) - \Phi_{\text{opt}}(z, c) - a(c)| \le \alpha_{\text{val}}, \tag{76}$$

and suppose the online refinement returns $\hat{z} \in U_c$ satisfying

$$\Phi_{\text{GDC}}(\hat{z}, c) \le \inf_{z\in U_c} \Phi_{\text{GDC}}(z, c) + \eta_{\text{opt}}. \tag{77}$$

If Φ_{opt} is m-strongly geodesically convex on U_c, then

$$\Phi_{\text{opt}}(\hat{z}, c) - \Phi_{\text{opt}}(z^\star, c) \le 2\alpha_{\text{val}} + \eta_{\text{opt}}, \qquad d_G(\hat{z}, z^\star) \le \sqrt{\frac{2(2\alpha_{\text{val}} + \eta_{\text{opt}})}{m}}. \tag{78}$$

Consequently, with a K_{D_ψ}-Lipschitz decoder, decoder error δ_{dec}, and final projection perturbation δ_{proj}, the executed first action obeys

$$\|u_t - u_t^\star\| \le K_{D_\psi} \sqrt{\frac{2(2\alpha_{\text{val}} + \eta_{\text{opt}})}{m}} + \delta_{\text{dec}} + \delta_{\text{proj}}. \tag{79}$$

Thus, along any sequence of datasets, models, and refinement budgets for which $\alpha_{\text{val}}, \eta_{\text{opt}}, \delta_{\text{dec}}$, and δ_{proj} vanish, GDC converges to the constrained optimal first action on the selected data-supported region.

Proof. Let $\widehat{\Phi} = \Phi_{\text{GDC}}(\cdot, c)$ and $\Phi^\star = \Phi_{\text{opt}}(\cdot, c)$. By (76), $\Phi^\star(\hat{z}) \le \widehat{\Phi}(\hat{z}) - a(c) + \alpha_{\text{val}}$. By (77), this is at most $\widehat{\Phi}(z^\star) - a(c) + \alpha_{\text{val}} + \eta_{\text{opt}}$. Applying (76) again at $z^\star$ gives $\Phi^\star(\hat{z}) - \Phi^\star(z^\star) \le 2\alpha_{\text{val}} + \eta_{\text{opt}}$. Strong geodesic convexity gives $\Phi^\star(\hat{z}) - \Phi^\star(z^\star) \ge (m/2) d_G(\hat{z}, z^\star)^2$, yielding the distance bound. The action bound follows from the decoder Lipschitz property, then adding the decoder reconstruction and final projection perturbation errors. Letting the four error terms go to zero proves the asymptotic statement. □

Corollary D.7 (Coverage-controlled rate)**.** *In the setting of Theorem D.6, let $\mathcal{Z}_n(c) \subset U_c$ be the finite latent training anchors for context c, with fill distance $h_n = \sup_{z\in U_c} \inf_{\zeta\in\mathcal{Z}_n(c)} d_G(z, \zeta)$. Suppose $\Phi_{\text{GDC}}(\cdot, c)$ and $\Phi_{\text{opt}}(\cdot, c)$ are Lipschitz on U_c with constants L_n and L_{opt}, and suppose their anchor error, up to a context normalizer $a(c)$, is at most ε_n:*

$$\sup_{\zeta\in\mathcal{Z}_n(c)} |\Phi_{\text{GDC}}(\zeta, c) - \Phi_{\text{opt}}(\zeta, c) - a(c)| \le \varepsilon_n.$$

Then Theorem D.6 holds with

$$\alpha_{\text{val}} \le \varepsilon_n + (L_n + L_{\text{opt}}) h_n.$$

Consequently, if the online refinement error after K refinement steps is η_K, the oracle gap is

$$\Phi_{\text{opt}}(\hat{z}, c) - \Phi_{\text{opt}}(z^\star, c) = \mathcal{O}(\varepsilon_n + h_n + \eta_K),$$

and the executed first-action error is

$$\|u_t - u_t^\star\| = \mathcal{O}\Big(\sqrt{\varepsilon_n + h_n + \eta_K} + \delta_{\mathrm{dec}} + \delta_{\mathrm{proj}}\Big).$$

For example, quasi-uniform coverage on a d-dimensional region gives $h_n = \mathcal{O}(n^{-1/d})$, while iid coverage with density bounded below gives $h_n = \mathcal{O}_p((\log n/n)^{1/d})$; the covering argument is given in Corollary E.25.

Proof. For any $z \in U_c$, choose $\zeta \in \mathcal{Z}_n(c)$ with $d_G(z,\zeta) \le h_n$. The triangle inequality and Lipschitz bounds give

$$|\Phi_{\mathrm{GDC}}(z,c) - \Phi_{\mathrm{opt}}(z,c) - a(c)| \le \varepsilon_n + (L_n + L_{\mathrm{opt}})h_n.$$

Substituting this bound for α_{val} in Theorem D.6 gives the value-gap and action-error rates. □

Remark D.8 (Physics–data interpolation)**.** *For $\lambda \in [0,1]$, the online potential*

$$\Phi_\lambda(z) = \lambda\Phi_{\mathrm{known}}(z) + (1-\lambda)\Phi_{\mathrm{data}}(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star) + \Phi_{\mathrm{task}}(z) \tag{80}$$

is affine in λ, so it interpolates continuously between known-physics control ($\lambda = 1$, leaving $\Phi_{\mathrm{known}} + \Phi_{\mathrm{task}}$) and data-driven progress completion ($\lambda = 0$, leaving $\Phi_{\mathrm{data}} + \Phi_{\mathrm{task}}$). The support restriction, anchors, filters, and final projection are those of the known scaffold at every λ, so intermediate values simply blend the two vector-field contributions. When data support is sparse or the optional support screen abstains, setting $\lambda \to 1$ removes Φ_{data} from the descent law and recovers the known-feasible controller.

These statements are intentionally conditional. GDC does not claim global optimality or implicit feasibility outside the data-supported scaffold. Approximate optimality requires the stronger value-consistency condition in Theorem D.6. Without that condition, the guarantee is narrower and operational: known physics enforces what can be executed, while progress-tilted feasible data supplies a local direction under the stated positive-progress assumption.

E Extended Theoretical Analysis

The following results expand the statements of Appendix D. Most are standard facts about Riemannian gradient flows and Langevin dynamics, specialized to the learned energy; they are included to fix notation and to make the modeling assumptions explicit, not as technical contributions. They analyze the fixed objects learned offline and used online: the empirical manifold, the progress-tilted density, its score field, and the induced latent control flow. These results are not additional offline training steps.

E.1 Notation and standing assumptions for the proofs

For the proof sections, write $c = (o, g, \mathcal{C}_{\mathrm{known}}, H^\star)$ for the complete conditioning context, including the selected horizon when it is available. All densities on $\widehat{\mathcal{M}}^*$ are densities with respect to the Riemannian volume measure dvol_G. Euclidean gradients in a latent chart are denoted by ∇_z, while grad_G denotes the Riemannian gradient. The corresponding Riemannian norm and dual norm are $\|v\|_G^2 = v^\top G(z) v$ and $\|\alpha\|_{G^{-1}}^2 = \alpha^\top G(z)^{-1}\alpha$. For two probability measures μ, ν on $\widehat{\mathcal{M}}^*$, $\|\mu - \nu\|_{\mathrm{TV}}$ denotes total variation distance with the convention $\|\mu - \nu\|_{\mathrm{TV}} = \sup_A |\mu(A) - \nu(A)| = \frac{1}{2}\|\mu - \nu\|_1$. The logarithmic score is the chart gradient $s_\theta = \nabla_z \log p_\theta^Z$; the associated Riemannian score vector is $G^{-1}s_\theta = \mathrm{grad}_G \log p_\theta^Z$.

Assumption E.1 (Standing regularity)**.** *The following conditions are used in the appendix whenever the corresponding object appears.*

(a) $\widehat{\mathcal{M}}^*$ is a connected C^2 finite-dimensional manifold equipped with a C^1 positive-definite metric G. The metric is uniformly elliptic on the data-supported region considered in the theorem. Either $\widehat{\mathcal{M}}^*$ has no boundary, or the boundary condition is reflecting/no-flux for all gradient-flow and diffusion arguments.

(b) The potentials Φ_{known}, Φ_{data}, Φ_{task}, and their sums are C^1 on the relevant region and bounded below; in results using Hessians or strong convexity they are C^2 on a geodesically convex neighborhood that does not intersect the cut locus of the minimizer.

(c) The conditional latent path densities $p_\theta^Z(\cdot \mid c)$ and their ideal targets are strictly positive and C^1 on the data-supported region. Score errors are measured in the dual Riemannian norm unless an explicit Euclidean constant is stated.

(d) The decoder $D_\psi(\cdot, c)$ is K_{D_ψ}-Lipschitz on the trusted latent region, and its reconstruction error for the first decoded action is bounded by δ_{dec}.

(e) The known-feasible execution set $\mathcal{U}_{\text{known}}(o_t, u_{t-1})$ is closed and nonempty whenever the projection layer is invoked. If no learned candidate is accepted, the fallback action is assumed to belong to this set.

(f) Progress certificates are bounded on every selected finite horizon: $|\Delta\rho(\tau, g)| \le R_\rho$. When clipped certificates are used, $R_\rho = c$.

Remark E.2 (Scope of the guarantees)**.** *The proofs are local to the data-supported known-feasible region where $H^\star \neq \varnothing$. Outside this region the algorithm may still execute a known-feasible fallback, but the progress-value and score-direction claims are not invoked.*

Remark E.3 (Frozen context)**.** *The stationary-distribution, free-energy, and Wasserstein statements, and every $\limsup_{t\to\infty}$ bound, are stated for a fixed conditioning context $c = (o, g, \mathcal{C}_{\text{known}}, H^\star)$. The deployed controller re-evaluates $H_t^\star$ and o_t at each step, so these results describe the frozen-context limit that the online iteration tracks between context changes, not a single time-homogeneous process.*

E.2 Energy-based interpretation and implicit value function

Data energy function

The data energy Φ_{data} already defined in Eq. (58) is exactly the negative log marginal likelihood $-\log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$ induced by the probabilistic path on $\widehat{\mathcal{M}}^*$; we reuse it, unchanged, as the energy function for the proofs below. Regions of low energy correspond to latent behaviors that are highly consistent with feasible trajectory demonstrations and their progress certificates under observation o, goal g, and the known constraint context.

Score function on the empirical manifold

Definition E.4 (Score function)**.** *The* score function *of the learned distribution on $\widehat{\mathcal{M}}^*$ is*

$$s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) = \nabla_z \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star) = -\nabla_z \Phi_{\text{data}}(z, o, g, \mathcal{C}_{\text{known}}, H^\star). \tag{81}$$

With a clean-behavior predictor, the score at noise level ℓ is induced by the predicted clean latent:

$$s_{\theta,\ell}^{\text{diff}}(z_\ell, o, g, \mathcal{C}_{\text{known}}, H^\star) \approx \frac{\sqrt{\bar\alpha_\ell}\, F_\theta(z_\ell, o, g, \mathcal{C}_{\text{known}}, H^\star, \ell) - z_\ell}{1 - \bar\alpha_\ell}. \tag{82}$$

Implicit progress value theorem

Local sensing defines the *feasible set* but not the *direction*: at an intersection, left, right, and straight may all be traversable, but not equally likely to reach the goal. A local controller $u_t = f(o_t)$ has access only to feasibility, not to directional intent.

GDC resolves this by replacing the local policy with a goal-conditioned latent trajectory distribution $\tau \sim p_\theta^\tau(\tau \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$, whose latent code z is selected and refined online. The key observation is that this distribution is trained on known-feasible trajectory segments with signed progress certificates. The objective upweights segments that improve a goal-conditioned progress certificate without requiring every short segment to be improving. Therefore, the distribution itself encodes directional information as a statistical bias: at the intersection, right-turning segments receive more

mass when they increase expected progress under similar contexts, while necessary waiting or repositioning behaviors need not be excluded.

This statistical bias is what the score function extracts: $s_\theta(z) = \nabla_z \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star)$ points toward latent regions whose decoded segments have higher probability under the progress-tilted conditional distribution. The known-constraint potential $\nabla\Phi_{\text{known}}(z)$ removes options that violate modeled physics; it does not choose among the remaining feasible options or recover unmodeled physics. The score does.

We now formalize this intuition.

Definition E.5 (Data-induced progress value)**.** *The* data-induced progress value *at latent point $z \in \widehat{\mathcal{M}}^*$ is*

$$V_{\text{prog}}(z, g) = -\log p_\theta^\tau(\textit{progress} \mid z, g, \mathcal{C}_{\text{known}}, H^\star), \tag{83}$$

where

$$p_\theta^\tau(\textit{progress} \mid z, g, \mathcal{C}_{\text{known}}, H^\star) = \int p_\theta^\tau(\tau \mid z, g, \mathcal{C}_{\text{known}}, H^\star)\, \mathbb{I}[\Delta\rho(\tau, g) > 0]\, \mathbb{I}[\tau \in \mathcal{C}_{\text{known}}]\, \mathrm{d}\tau \tag{84}$$

is the probability of generating a known-feasible trajectory segment that improves the progress certificate ρ.

Proposition E.6 (Score as progress-value gradient)**.** *Fix a context $c_0 = (o, g, \mathcal{C}_{\text{known}}, H^\star)$ satisfying Assumption E.1, and write $H := H^\star$. Let $\Gamma_H(z, c_0)$ be the set of known-feasible horizon-H continuations starting from latent point z, with reference measure ν_z; a continuation τ carries the latent rollout $(z_j)_{j<H}$ used to evaluate C_H. Under the maximum-entropy model of feasible-progress demonstrations, suppose a trajectory continuation has unnormalized weight*

$$\exp(-C_H(\tau; c_0)), \qquad C_H(\tau; c_0) = \sum_{j=0}^{H-1} \ell_{\text{prog}}(z_j, u_j; c_0), \tag{85}$$

where lower ℓ_{prog} corresponds to larger signed progress certificate improvement. Assume the partition function below is finite, positive, and C^1 in z on the data-supported region. Define the soft progress value

$$V_{\text{prog}}^{\text{soft}}(z, c_0) = -\log Z_H(z, c_0), \qquad Z_H(z, c_0) = \int_{\Gamma_H(z, c_0)} \exp[-C_H(\tau; c_0)]\, \mathrm{d}\nu_z(\tau). \tag{86}$$

If the ideal progress-tilted latent density is $p^\star(z \mid c_0) = Z_{c_0}^{-1} Z_H(z, c_0)$, then

$$-\log p^\star(z \mid c_0) = V_{\text{prog}}^{\text{soft}}(z, c_0) + \log Z_{c_0}. \tag{87}$$

This identification is a modeling assumption: it holds when the empirical progress-weighted aggregate posterior q_β^Z of (46)*, whose reference measure is the known-feasible data measure, agrees with the maximum-entropy continuation density $Z_{c_0}^{-1} Z_H(\cdot, c_0)$; the theorem does not derive this agreement from the offline objective. Consequently, if the learned latent path has C^1 log-density error $\|\nabla_z \log p_\theta^Z - \nabla_z \log p^\star\| \le \varepsilon_{\text{approx}}$ on the data-supported region, then*

$$\boxed{s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) = \nabla_z \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star) = -\nabla_z V_{\text{prog}}^{\text{soft}}(z, c_0) + \mathcal{O}(\varepsilon_{\text{approx}}),} \tag{88}$$

Proof. By definition of $p^\star$,

$$\log p^\star(z \mid c_0) = \log Z_H(z, c_0) - \log Z_{c_0} = -V_{\text{prog}}^{\text{soft}}(z, c_0) - \log Z_{c_0}.$$

The normalizer Z_{c_0} depends only on the conditioning context, not on z. Differentiating with respect to z therefore gives

$$\nabla_z \log p^\star(z \mid c_0) = -\nabla_z V_{\text{prog}}^{\text{soft}}(z, c_0).$$

The assumed C^1 regularity of Z_H justifies the differentiation; one sufficient condition is a smoothly parameterized continuation set and dominated differentiation under the reference measures ν_z. Finally, $s_\theta = \nabla_z \log p_\theta^Z$ and the assumed C^1 log-density approximation imply

$$\left\| s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) + \nabla_z V_{\text{prog}}^{\text{soft}}(z, c_0) \right\| \le \varepsilon_{\text{approx}},$$

which is (88). □

Remark E.7 (Resolving directional ambiguity)**.** *Proposition E.6 states the condition under which GDC resolves the directional ambiguity problem. The mechanism is that training with signed progress certificates creates a distribution that favors larger signed progress in the training measure. Positive expected progress for a selected horizon is the stated assumption, not a consequence of the support-screen statistic* (27)*.*

At a decision point (e.g., an intersection), the observation o_t defines the feasible set $\{$left, right, straight$\}$ but provides no directional preference. The path model $p_\theta^\tau(\tau \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$, trained with progress-weighted feasible segments, places relatively more mass on directions with larger progress certificates in similar supported contexts. The score $s_\theta = \nabla_z \log p_\theta^Z$ converts this statistical bias into a deterministic gradient of the learned density. The known constraint potential $\nabla\Phi_{\text{known}}$ removes explicitly modeled infeasible options; the score chooses among known-feasible options and approximates the implicit feasibility structure.

Concretely, define $V_{\text{prog}}(z, g) = -\log p_\theta^\tau(\text{progress} \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$. If the path model is trained with signed progress certificates and the selected context satisfies the positive-progress assumption, the distribution has the progress-bias property*:*

$$\mathbb{E}_{\tau \sim p_\theta^\tau(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star)}[\Delta\rho(\tau, g)] > 0. \tag{89}$$

The short-horizon distribution is biased toward improving the progress certificate in expectation; individual samples may still wait, backtrack, or reposition when such behavior appears in the data-supported strategy. The directional information is compressed into the distribution over short trajectories, because the training objective weights feasible segments by progress. The online controller ranks short candidate segments from its horizon pool; the optional screen may narrow that pool when empirical support is sparse.

Remark E.8 (Scope of the implicit value)**.** *The implicit $V_{\text{prog}}(z, g)$ is a progress value of the* data policy*, not the optimal value function. It encodes "how likely is local progress if you behave like the demonstrations," not "how likely is final success under the best possible behavior." The Boltzmann-rational assumption* (85) *is an idealization; real demonstrations are only approximately maximum-entropy. The connection is strongest when training data is diverse and covers the relevant state space.*

TOTAL CONTROL POTENTIAL ON THE EMPIRICAL MANIFOLD

Let $\Phi_{\text{task}} : \widehat{\mathcal{M}}^* \to \mathbb{R}$ encode task-specific objectives.

Definition E.9 (Total control potential)**.** *The* total control potential *on $\widehat{\mathcal{M}}^*$ is*

$$\begin{aligned} \Phi_{\text{GDC}}(z) &= \Phi_{\text{known}}(z) + \Phi_{\text{data}}(z, o, g, \mathcal{C}_{\text{known}}, H^\star) \\ &\quad + \Phi_{\text{task}}(z) \\ &= \Phi_{\text{known}}(z) - \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star) \\ &\quad + \Phi_{\text{task}}(z). \end{aligned} \tag{90}$$

By Proposition E.6, this is approximately, up to a z-independent constant:

$$\Phi_{\text{GDC}}(z) \approx \Phi_{\text{known}}(z) + V_{\text{prog}}^{\text{soft}}(z, c_0) + \Phi_{\text{task}}(z), \tag{91}$$

so control minimizes the sum of explicit known physics, the data-induced approximation of implicit information, and the task cost.

E.3 SCORE-GUIDED CONTROL AS RIEMANNIAN GRADIENT FLOW

RIEMANNIAN GRADIENT FLOW ON THE EMPIRICAL MANIFOLD

Rather than performing gradient flow in flat Euclidean space, GDC implements control as *Riemannian* gradient flow on $\widehat{\mathcal{M}}^*$, respecting the intrinsic geometry of the learned behavior manifold while explicitly enforcing known constraints.

Definition E.10 (Riemannian gradient flow control)**.** *The* Riemannian gradient flow control dynamics *on $\widehat{\mathcal{M}}^*$ are*

$$\dot{z} = -\operatorname{grad}_G \Phi_{\text{GDC}}(z) = -G(z)^{-1} \nabla_z \Phi_{\text{GDC}}(z). \tag{92}$$

Proposition E.11 (Score-guided Riemannian control)**.** *Under the decomposition* (90)*, the Riemannian gradient flow* (92) *is equivalent to*

$$\dot{z} = G(z)^{-1}[s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) - \nabla_z \Phi_{\text{known}}(z) - \nabla_z \Phi_{\text{task}}(z)]\,. \tag{93}$$

Proof. Compute:

$$\begin{aligned} \operatorname{grad}_G \Phi_{\text{GDC}}(z) &= G(z)^{-1} \nabla_z \Phi_{\text{GDC}}(z) && (94)\\ &= G(z)^{-1}[-s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) + \nabla_z \Phi_{\text{known}}(z) + \nabla_z \Phi_{\text{task}}(z)]\,. && (95) \end{aligned}$$

Substituting into $\dot{z} = -\operatorname{grad}_G \Phi_{\text{GDC}}(z)$ yields (93). □

Remark E.12 (From sampling to gradient flow)**.** *The mathematical distinction between direct generative policy sampling and GDC is precisely the sampling-versus-energy distinction stated in Section 1. In diffusion policy, the score drives a* sampling *process: $\dot{u} = s_\theta(u)$, where the dynamics* are *the controller. In GDC, the score enters a* structured control law*:*

$$\dot{z} = G(z)^{-1}[s_\theta(z, o, g, \mathcal{C}_{\text{known}}, H^\star) - \nabla_z \Phi_{\text{known}}(z) - \nabla_z \Phi_{\text{task}}(z)]\,, \tag{96}$$

where the Riemannian metric $G(z)$ respects manifold curvature, Φ_{known} encodes modeled physics and known constraints, and Φ_{task} encodes task objectives. Known constraints are enforced by the support restriction $\widehat{\mathcal{M}}^*_{\text{known}}$, anchor penalties, and candidate filtering. The score provides the implicit information (via the implicit value gradient, Proposition E.6); the known scaffold provides explicit feasibility and stability.

In short: diffusion policy performs sampling in action space; GDC defines a controlled Riemannian gradient flow where the path score provides direction and geometry enforces feasibility.

Singular perturbation structure for known constraints

The online potential $\Phi_{\text{on},t}$ and its Riemannian descent field F_t already defined in Eqs. (57)–(59) admit a singular perturbation reading. Writing $\Phi_{\text{scaf}}(z) = d_G(z, \mathcal{M}_t)^2$ for the squared distance to the local known-feasible set:

- Φ_{scaf} is the fast scaffold potential, entering at $O(\epsilon_{\text{s}}^{-1})$;
- Φ_{known}, Φ_{data}, and Φ_{task} are all tangential potentials entering at the same ordinary $O(1)$ slow scale, weighted by λ, $1-\lambda$, and 1 respectively.

This is exactly the two-timescale structure of Section 3.3 in the main text: one singular fast term and three ordinary slow terms, not four separate rates.

Discrete-time implementation

For computational implementation, the flow (59) is discretized on the learned map:

$$z_{t+1} = z_t \oplus \big(\dot{z}_{\text{nom},t}\, \Delta t\big), \tag{97}$$

where $\oplus$ denotes ordinary addition in a flat latent chart or a standard retraction for curved coordinates. Feasibility is maintained by selecting and refining within the anchored support $\widehat{\mathcal{M}}^*_{\text{known}}(o, g)$ and by the known-constraint potential Φ_{known}.

E.4 Langevin control dynamics on the empirical manifold

Manifold Langevin SDE

The stochastic analysis in this and the next subsection is not used by the deterministic controller in the experiments; it describes the frozen-context behavior of a sampling variant. The deterministic Riemannian gradient flow (92) is augmented with a stochastic noise term to obtain the *manifold Langevin SDE*:

$$\mathrm{d}z_t = -\operatorname{grad}_G \Phi_{\text{GDC}}(z_t)\,\mathrm{d}t + \sqrt{2\beta_{\text{L}}^{-1}}\, G(z_t)^{-1/2}\,\mathrm{d}W_t, \tag{98}$$

where W_t is a standard Brownian motion in $\mathbb{R}^{d_z}$, $\beta_{\mathrm{L}} > 0$ is an inverse temperature parameter (a sampling knob, unrelated to the offline progress tilt β), and $G(z)^{-1/2}$ ensures the noise respects the Riemannian geometry. Equation (98) is an intrinsic shorthand: the diffusion is the Riemannian Langevin process whose generator acts on smooth test functions φ as

$$\mathcal{L}\varphi = -\langle \mathrm{grad}_G \, \Phi_{\mathrm{GDC}}, \, \mathrm{grad}_G \, \varphi \rangle_G + \beta_{\mathrm{L}}^{-1} \Delta_G \varphi. \tag{99}$$

In local coordinates this generator includes the usual metric/Christoffel drift terms associated with Brownian motion on $(\widehat{\mathcal{M}}^*, G)$; the compact notation above suppresses those coordinate artifacts.

Stochastic score-guided control on the empirical manifold

Incorporating the score decomposition, the stochastic control dynamics on $\widehat{\mathcal{M}}^*$ use $H_t^\star = H^\star(o_t, g, \mathcal{C}_{\mathrm{known}})$ and are:

$$\begin{aligned} \mathrm{d}z_t = G(z_t)^{-1} [s_\theta(z_t, o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) - \nabla \Phi_{\mathrm{known}}(z_t) - \nabla \Phi_{\mathrm{task}}(z_t)] \, \mathrm{d}t \\ + \sqrt{2\beta_{\mathrm{L}}^{-1}} \, G(z_t)^{-1/2} \, \mathrm{d}W_t. \end{aligned} \tag{100}$$

The noise term plays the role of stochastic exploration on $\widehat{\mathcal{M}}^*$: at high temperature ($\beta_{\mathrm{L}} \to 0$), the dynamics explore broadly across behavior modes on the manifold; at low temperature ($\beta_{\mathrm{L}} \to \infty$), the dynamics reduce to the deterministic Riemannian gradient flow.

Euler–Maruyama discretization

The stochastic dynamics can be discretized by an Euler–Maruyama step in any chosen latent chart:

$$\begin{aligned} v_t = \Delta t \, G(z_t)^{-1} [s_\theta(z_t, o_t, g, \mathcal{C}_{\mathrm{known}}, H_t^\star) - \nabla \Phi_{\mathrm{known}}(z_t) - \nabla \Phi_{\mathrm{task}}(z_t)] \\ + \sqrt{2\Delta t \beta_{\mathrm{L}}^{-1}} \, G(z_t)^{-1/2} \, \xi_t, \end{aligned} \tag{101}$$

$$z_{t+1} = z_t + v_t, \tag{102}$$

where $\xi_t \sim \mathcal{N}(0, I)$. If a particular implementation uses group-valued coordinates, the additive update can be replaced by the corresponding retraction, but this is not a core component of GDC.

E.5 Fokker–Planck analysis on the empirical manifold

The Fokker–Planck equation on the manifold

Let $p(z, t)$ denote the probability density of the process (100) at time t with respect to the Riemannian volume form on $\widehat{\mathcal{M}}^*$. The evolution satisfies:

$$\frac{\partial p}{\partial t} = \mathrm{div}_G(p \, \mathrm{grad}_G \, \Phi_{\mathrm{GDC}}) + \beta_{\mathrm{L}}^{-1} \, \Delta_G \, p, \tag{103}$$

where div_G and Δ_G are the Riemannian divergence and Laplace–Beltrami operators on $(\widehat{\mathcal{M}}^*, G)$.

Stationary distribution

Proposition E.13 (Stationary distribution on $\widehat{\mathcal{M}}^*$)**.** *Under Assumption E.1, suppose* $Z = \int_{\widehat{\mathcal{M}}^*} \exp(-\beta_{\mathrm{L}} \Phi_{\mathrm{GDC}}(z)) \, \mathrm{dvol}_G(z) < \infty$. *Then the manifold Langevin control dynamics have the Gibbs stationary distribution*

$$\pi^\star(z) = \frac{1}{Z} \exp(-\beta_{\mathrm{L}} \, \Phi_{\mathrm{GDC}}(z)), \qquad Z = \int_{\widehat{\mathcal{M}}^*} \exp(-\beta_{\mathrm{L}} \, \Phi_{\mathrm{GDC}}(z)) \, \mathrm{dvol}_G(z). \tag{104}$$

If the data-supported region is connected and the diffusion is uniformly elliptic and nonexplosive there, this stationary distribution is unique; in particular, this holds on a compact connected region with reflecting/no-flux boundary.

Proof. The Fokker–Planck equation can be written as a conservation law $\partial_t p = -\operatorname{div}_G J_p$ with probability flux

$$J_p = -p \operatorname{grad}_G \Phi_{\mathrm{GDC}} - \beta_{\mathrm{L}}^{-1} \operatorname{grad}_G p.$$

For $\pi^\star = Z^{-1} \exp(-\beta_{\mathrm{L}} \Phi_{\mathrm{GDC}})$,

$$\operatorname{grad}_G \pi^\star = -\beta_{\mathrm{L}} \pi^\star \operatorname{grad}_G \Phi_{\mathrm{GDC}}.$$

Substitution gives $J_{\pi^\star} = 0$, so $\partial_t \pi^\star = 0$ and the reflecting boundary condition, when a boundary is present, is automatically satisfied. Existence follows from the assumed finiteness of Z. For uniqueness, let p be any smooth invariant density in the same no-flux domain and write $p = h\pi^\star$. Since the generator (99) is reversible with respect to $\pi^\star$, the stationarity equation tested against h gives $\int_{\widehat{\mathcal{M}}^*} \|\operatorname{grad}_G h\|_G^2 \, \pi^\star \operatorname{dvol}_G = 0$. Hence h is constant on each connected component. Connectedness and normalization force $h \equiv 1$, so the stationary density is unique. □

Implications for control

The stationary distribution reveals:

$$\pi^\star(z) \propto \exp\big(\beta_{\mathrm{L}} \log p_\theta^Z(z \mid o, g, \mathcal{C}_{\mathrm{known}}, H^\star)\big) \cdot \exp(-\beta_{\mathrm{L}} \Phi_{\mathrm{known}}(z)) \cdot \exp(-\beta_{\mathrm{L}} \Phi_{\mathrm{task}}(z)), \quad (105)$$

which is the distribution over behaviors on $\widehat{\mathcal{M}}^*$ that balances data consistency, known-constraint satisfaction, and task optimality.

Free energy and convergence

Define the free energy functional on $\widehat{\mathcal{M}}^*$:

$$\mathcal{F}[p] = \mathbb{E}_p[\Phi_{\mathrm{GDC}}] - \beta_{\mathrm{L}}^{-1} H_G(p), \quad (106)$$

where $H_G(p) = -\int_{\widehat{\mathcal{M}}^*} p \log p \operatorname{dvol}_G$ is the differential entropy with respect to the Riemannian volume form.

Proposition E.14 (Free energy descent on $\widehat{\mathcal{M}}^*$)**.** *For smooth positive solutions p_t of the Fokker–Planck evolution* (103) *satisfying the no-flux boundary condition when a boundary is present,*

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \mathcal{F}[p_t] &= -\int_{\widehat{\mathcal{M}}^*} p_t \left\| \operatorname{grad}_G\big(\Phi_{\mathrm{GDC}} + \beta_{\mathrm{L}}^{-1} \log p_t\big) \right\|_G^2 \operatorname{dvol}_G \\ &= -\beta_{\mathrm{L}}^{-2} \int_{\widehat{\mathcal{M}}^*} p_t \left\| \operatorname{grad}_G \log \frac{p_t}{\pi^\star} \right\|_G^2 \operatorname{dvol}_G \le 0. \end{aligned} \quad (107)$$

Proof. Using $H_G(p) = -\int p \log p \operatorname{dvol}_G$, the free energy is

$$\mathcal{F}[p] = \int_{\widehat{\mathcal{M}}^*} p \Phi_{\mathrm{GDC}} \operatorname{dvol}_G + \beta_{\mathrm{L}}^{-1} \int_{\widehat{\mathcal{M}}^*} p \log p \operatorname{dvol}_G.$$

Its first variation is

$$\frac{\delta \mathcal{F}}{\delta p} = \Phi_{\mathrm{GDC}} + \beta_{\mathrm{L}}^{-1} (\log p + 1).$$

The additive constant has zero gradient. The Fokker–Planck equation is equivalently

$$\partial_t p = \operatorname{div}_G\big(p \operatorname{grad}_G \big(\Phi_{\mathrm{GDC}} + \beta_{\mathrm{L}}^{-1} \log p\big)\big).$$

Therefore, integrating by parts on a boundaryless manifold or using the no-flux boundary condition,

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathcal{F}[p_t] = \int_{\widehat{\mathcal{M}}^*} \frac{\delta \mathcal{F}}{\delta p} \partial_t p_t \operatorname{dvol}_G = -\int_{\widehat{\mathcal{M}}^*} p_t \left\| \operatorname{grad}_G\big(\Phi_{\mathrm{GDC}} + \beta_{\mathrm{L}}^{-1} \log p_t\big) \right\|_G^2 \operatorname{dvol}_G.$$

Since $\log(p_t/\pi^\star) = \log p_t + \beta_{\mathrm{L}} \Phi_{\mathrm{GDC}} + \log Z$, its gradient is $\beta_{\mathrm{L}} \operatorname{grad}_G(\Phi_{\mathrm{GDC}} + \beta_{\mathrm{L}}^{-1} \log p_t)$, which gives the second identity and nonpositivity. □

E.6 TRAJECTORY-LEVEL PROBABILISTIC PATH CONTROL

RECEDING-HORIZON TRAJECTORY CONTROL

When inter-step constraints couple multiple time steps, trajectory-level planning on the empirical manifold is preferable. This section extends GDC to the trajectory manifold $\widehat{\mathcal{M}}^{H,*}$: known dynamics and constraints define the scaffold, while feasible trajectory data with progress certificates complete the implicit trajectory-level coupling structure.

TRAJECTORY PATH MODEL ON THE EMPIRICAL HORIZON MANIFOLD

Define a trajectory of actions over horizon H: $\tau = (u_t, \dots, u_{t+H-1}) \in \mathcal{U}^H$. The empirical trajectory manifold $\widehat{\mathcal{M}}^{H,*}$ is constructed by encoding and decoding feasible horizon action sequences, while known dynamics are used only where they are available to check or penalize decoded sequences:

$$\mathcal{L}_{\text{horizon-known}} = \frac{1}{N} \sum_{k=1}^{N} \sum_{r} \left[f_r^{\text{known}}\big(D_\psi^H(z_k, o_k, g_k, \mathcal{C}_{\text{known}}), o_k, g_k\big)_+ \right]^2, \tag{108}$$

where D_ψ^H decodes an H-step action sequence. If a known rollout map is available, f_r^{known} may include its dynamic-consistency residuals; if not, it includes only the available safety, actuator, timing, or resource constraints. Residual couplings not captured by f_{known} are learned from feasible trajectory data through Φ_{data}.

We train a conditional probabilistic path model on $\widehat{\mathcal{M}}^{H,*}$:

$$p_\theta^\tau(\tau \mid o_t, g, \mathcal{C}_{\text{known}}, H^\star) \tag{109}$$

with the same clean-latent prediction objective (43), replacing z with the trajectory latent code on $\widehat{\mathcal{M}}^{H,*}$ and using D_ψ^H to decode the selected clean latent code into an action sequence.

TRAJECTORY ENERGY AND SCORE

Definition E.15 (Trajectory energy on $\widehat{\mathcal{M}}^{H,*}$)**.**

$$\Phi_\tau(\tau) = -\log p_\theta^\tau(\tau \mid o_t, g, \mathcal{C}_{\text{known}}, H^\star) + \Phi_{\text{known}}(\tau) + \Phi_{\text{task}}(\tau), \tag{110}$$

where $\Phi_{\text{known}}(\tau)$ encodes trajectory-level known constraints and $\Phi_{\text{task}}(\tau)$ encodes trajectory-level objectives.

Let $s_\theta^\tau(\tau, o_t, g, \mathcal{C}_{\text{known}}, H^\star) = \nabla_\tau \log p_\theta^\tau(\tau \mid o_t, g, \mathcal{C}_{\text{known}}, H^\star)$. The score-guided trajectory dynamics on $\widehat{\mathcal{M}}^{H,*}$ are:

$$\dot{\tau} = G_\tau(\tau)^{-1}[s_\theta^\tau(\tau, o_t, g, \mathcal{C}_{\text{known}}, H^\star) - \nabla_\tau \Phi_{\text{known}}(\tau) - \nabla_\tau \Phi_{\text{task}}(\tau)], \tag{111}$$

where G_τ is the Riemannian metric on the trajectory manifold.

RECEDING HORIZON EXECUTION

Only the first action is executed: $u_t = \tau_0^\star$, where $\tau^\star = \arg\min_\tau \Phi_\tau(\tau)$ on $\widehat{\mathcal{M}}^{H,*}$. The horizon shifts and the process repeats, yielding a receding-horizon controller that enforces known trajectory-level constraints and uses the path model's distributional knowledge to approximate implicit trajectory information.

PROGRESS CONSISTENCY ACROSS HORIZON

The trajectory-level formulation gives a simple sufficient condition under which the learned distribution retains a progress bias. Define the implicit progress value $V_{\text{prog}}(z, g) = -\log p_\theta^\tau(\text{progress} \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$ as in Definition E.5.

Proposition E.16 (Progress bias under trajectory sampling). *Let $q_\beta(\tau \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$ be the progress-tilted known-feasible target distribution at a selected horizon. Suppose that, on a data-supported region $U \subset \widehat{\mathcal{M}}^*$, there is a margin $\gamma > 0$ such that*

$$\mathbb{E}_{q_\beta(\cdot|z,g,\mathcal{C}_{\text{known}},H^\star)}[\Delta\rho(\tau, g)] \geq \gamma \qquad \text{for all } z \in U.$$

Suppose also that the learned trajectory path satisfies

$$\|p_\theta^\tau(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star) - q_\beta(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star)\|_{\text{TV}} \leq \delta \qquad \text{for all } z \in U,$$

with $2R_\rho\delta < \gamma$. Then, for latent points within U,

$$\boxed{\mathbb{E}_{\tau\sim p_\theta^\tau(\cdot|z,g,\mathcal{C}_{\text{known}},H^\star)}[\Delta\rho(\tau, g)] > 0.} \tag{112}$$

That is, the learned short-horizon trajectory distribution is biased toward known-feasible progress in expectation.

Proof. Fix $z \in U$ and abbreviate $p = p_\theta^\tau(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$ and $q = q_\beta(\cdot \mid z, g, \mathcal{C}_{\text{known}}, H^\star)$. Since $|\Delta\rho(\tau, g)| \leq R_\rho$ by Assumption E.1, the total-variation variational characterization gives

$$|\mathbb{E}_p[\Delta\rho(\tau, g)] - \mathbb{E}_q[\Delta\rho(\tau, g)]| \leq 2R_\rho\|p - q\|_{\text{TV}} \leq 2R_\rho\delta.$$

Therefore

$$\mathbb{E}_p[\Delta\rho(\tau, g)] \geq \mathbb{E}_q[\Delta\rho(\tau, g)] - 2R_\rho\delta \geq \gamma - 2R_\rho\delta > 0.$$

The conclusion is conditional on data support and on the fitting error bound; it does not assert global optimality or final-goal success for every sampled short segment. □

Remark E.17 (Short horizons and directional ambiguity). *Equation* (112) *says that, even though each planned trajectory is only H steps long, the* distribution *over such trajectories is biased toward local progress. The directional information is not computed by planning over a long horizon; it is compressed into the* distribution *over short trajectories because the training objective uses signed progress certificates. At the intersection, if right-turning snippets have higher expected progress under similar goals and contexts, the distribution concentrates more mass on right-turning short trajectories while still allowing waiting or yielding behavior when supported by the data.*

The learned distribution can supply a useful local direction when the stated progress-bias condition holds, even with short candidate segments. This does not imply that every sampled segment reaches the final goal.

E.7 Stability analysis

Lyapunov stability of Riemannian gradient flow

Proposition E.18 (Lyapunov stability on $\widehat{\mathcal{M}}^*$; standard). *Let $\Phi_{\text{GDC}} : \widehat{\mathcal{M}}^* \to \mathbb{R}$ be continuously differentiable and bounded below, with $z^\star$ an interior strict local minimum of the data-supported region. Consider the Riemannian gradient flow* (92)*. Then:*

- *(i) $z^\star$ is a Lyapunov stable equilibrium on $\widehat{\mathcal{M}}^*$.*
- *(ii) Every trajectory in a sublevel set $\{z \in \widehat{\mathcal{M}}^* : \Phi_{\text{GDC}}(z) \leq c\}$ remains in that set.*
- *(iii) If Φ_{GDC} is m-strongly geodesically convex near $z^\star$, then $z^\star$ is exponentially stable with rate m:*

$$d_G(z_t, z^\star) \leq e^{-mt}\, d_G(z_0, z^\star), \tag{113}$$

where d_G is the geodesic distance on $(\widehat{\mathcal{M}}^*, G)$.

Proof. Since $z^\star$ is an interior strict local minimum and Φ_{GDC} is continuous, there is a neighborhood U of $z^\star$ such that $V(z) = \Phi_{\text{GDC}}(z) - \Phi_{\text{GDC}}(z^\star)$ is positive on $U \setminus \{z^\star\}$ and $V(z^\star) = 0$. Since $z^\star$ is an interior local minimum of a C^1 function on the local chart, $\text{grad}_G\, \Phi_{\text{GDC}}(z^\star) = 0$, so $z^\star$ is an equilibrium. Along the Riemannian gradient flow,

$$\frac{\mathrm{d}}{\mathrm{d}t}V(z_t) = \mathrm{d}\Phi_{\text{GDC}}(z_t)[\dot{z}_t] = \langle \text{grad}_G\, \Phi_{\text{GDC}}(z_t),\, \dot{z}_t\rangle_G = -\,\|\text{grad}_G\, \Phi_{\text{GDC}}(z_t)\|_G^2 \leq 0. \tag{114}$$

This proves Lyapunov stability by the standard Lyapunov sublevel argument: for any sufficiently small neighborhood U_0 of $z^\star$, choose a compact sublevel set of V contained in U_0; the trajectory cannot leave it because V is nonincreasing. The same calculation proves (ii), since every sublevel set of Φ_{GDC} is forward invariant.

For (iii), restrict to a geodesically convex normal neighborhood in which the squared distance is smooth. Let $W(t) = \frac{1}{2} d_G(z_t, z^\star)^2$. The first-variation formula gives

$$\dot{W}(t) = \langle -\mathrm{Log}_{z_t}(z^\star),\, \dot{z}_t \rangle_G = \langle \mathrm{Log}_{z_t}(z^\star),\, \mathrm{grad}_G\, \Phi_{\mathrm{GDC}}(z_t) \rangle_G.$$

Geodesic m-strong convexity with minimizer $z^\star$ implies $\langle \mathrm{grad}_G\, \Phi_{\mathrm{GDC}}(z_t),\, -\mathrm{Log}_{z_t}(z^\star) \rangle_G \geq m\, d_G(z_t, z^\star)^2$. Hence $\dot{W}(t) \leq -m\, d_G(z_t, z^\star)^2 = -2mW(t)$. Gronwall's inequality gives $W(t) \leq e^{-2mt} W(0)$, and therefore $d_G(z_t, z^\star) \leq e^{-mt} d_G(z_0, z^\star)$. □

Robustness to score estimation error

Define the score estimation error $\delta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star) = s_\theta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star) - s^\star(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star)$.

Assumption E.19 (Bounded score error)**.** *The score error is uniformly bounded in the dual Riemannian norm:*

$$\sup_{z \in \widehat{\mathcal{M}}^*, o, g} \|\delta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star)\|_{G^{-1}} \leq \varepsilon.$$

Proposition E.20 (Robustness on $\widehat{\mathcal{M}}^*$)**.** *Under Assumption E.19, suppose the true total potential is m-strongly geodesically convex on a geodesically convex data-supported neighborhood containing the trajectory and has minimizer $z^\star$. Then the approximate dynamics converge to a neighborhood:*

$$\limsup_{t \to \infty} d_G(z_t, z^\star) \leq \frac{\varepsilon}{m}. \tag{115}$$

Proof. Let $e(z) = G(z)^{-1} \delta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star)$, so $\|e(z)\|_G = \|\delta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star)\|_{G^{-1}} \leq \varepsilon$. In the geodesically convex neighborhood, use $V(z) = \frac{1}{2} d_G(z, z^\star)^2$. The perturbed Riemannian gradient flow is $\dot{z} = -\mathrm{grad}_G\, \Phi^\star(z) + e(z)$. By the first-variation formula,

$$\begin{aligned} \dot{V} &= \langle -\mathrm{Log}_z(z^\star),\, -\mathrm{grad}_G\, \Phi^\star(z) + e(z) \rangle_G \\ &= \langle \mathrm{Log}_z(z^\star),\, \mathrm{grad}_G\, \Phi^\star(z) \rangle_G - \langle \mathrm{Log}_z(z^\star),\, e(z) \rangle_G \\ &\leq -m\, d_G(z, z^\star)^2 + \varepsilon\, d_G(z, z^\star) \\ &= -d_G(z, z^\star)\, (m\, d_G(z, z^\star) - \varepsilon)\,. \end{aligned} \tag{116}$$

Thus $\dot{V} < 0$ whenever $d_G(z, z^\star) > \varepsilon/m$. For every $\eta > 0$, the closed exterior of the ball of radius $\varepsilon/m + \eta$ has strictly negative derivative except possibly at its boundary, so trajectories cannot remain outside that enlarged ball indefinitely. Letting $\eta \downarrow 0$ gives the stated limsup bound. □

Non-accumulation of discretization drift

Local refinement introduces a small discretization drift per step. The next statement records that the retraction/backtracking guard of the online controller is what keeps this local integration error uniformly controlled; it is a design guarantee of that guard, not a consequence of the integrator alone.

Proposition E.21 (Non-accumulation of drift)**.** *Let $\Phi_{\mathrm{scaf}}(z) = d_G(z, \mathcal{M}_t)^2$, where $\mathcal{M}_t$ is the local known-feasible latent scaffold. Suppose each accepted latent update starts in $\mathcal{M}_t$, has capped metric length at most $V_{\max} \Delta t$, and applies the retraction/backtracking guard $d_G(z_{t+1}, \mathcal{M}_t) \leq \sqrt{\epsilon_{\mathrm{s}}/2}\, V_{\max} \Delta t$. Then the manifold deviation satisfies, for every t,*

$$\Phi_{\mathrm{scaf}}(z_t) \leq \frac{\epsilon_{\mathrm{s}}}{2}\, V_{\max}^2\, \Delta t^2. \tag{117}$$

The bound is uniform in t, vanishes as $\Delta t \to 0$, and does not accumulate.

Proof. The capped integrator guarantees that the raw update moves by at most $V_{\max}\Delta t$ in the metric. The scaffold correction is accepted only when the post-correction point has distance at most $\sqrt{\epsilon_{\mathrm{s}}/2}\, V_{\max}\Delta t$ from $\mathcal{M}_t$. Since Φ_{scaf} is the squared distance to $\mathcal{M}_t$,

$$\Phi_{\mathrm{scaf}}(z_t) = d_G(z_t, \mathcal{M}_t)^2 \le \frac{\epsilon_{\mathrm{s}}}{2} V_{\max}^2 \Delta t^2.$$

This is a per-step guard on the accepted latent state rather than a sum of local truncation errors. Hence the same upper bound holds uniformly over time and cannot grow with the rollout horizon. □

Action error under an explicit tracking condition

Proposition E.22 (Conditional first-action error)**.** *Fix a context and a feasible ideal latent point $z_t^\star$. Suppose the accepted numerical point z_t lies within $\sqrt{\epsilon_{\mathrm{s}}/2}\, V_{\max}\Delta t$ of the particular in-scaffold refinement point $\bar{z}_t$, and $d_G(\bar{z}_t, z_t^\star) \le \varepsilon/m$. If the decoder is K_{D_ψ}-Lipschitz, its first-action reconstruction error is at most δ_{dec}, and final projection is nonexpansive and fixes the ideal action, then the executed first-action error satisfies*

$$\|u_t - u_t^\star\| \le K_{D_\psi}\left(\sqrt{\frac{\epsilon_{\mathrm{s}}}{2}}\, V_{\max}\, \Delta t + \frac{\varepsilon}{m}\right) + \delta_{dec}. \tag{118}$$

The scaffold-deviation bound alone does not establish the tracking condition.

Proof. By the two stated tracking conditions and the triangle inequality, $d_G(z_t, z_t^\star) \le \sqrt{\epsilon_{\mathrm{s}}/2}\, V_{\max}\Delta t + \varepsilon/m$. The decoder is K_{D_ψ}-Lipschitz on this neighborhood, so the first decoded actions satisfy

$$\|D_\psi(z_t, c)_0 - D_\psi(z_t^\star, c)_0\| \le K_{D_\psi} d_G(z_t, z_t^\star).$$

Adding the decoder reconstruction error δ_{dec} gives (118). The final known-constraint projection onto the convex set $\mathcal{U}_{\mathrm{known}}(o_t, u_{t-1})$ is nonexpansive and leaves the feasible ideal action fixed, so it does not enlarge the bound; when $\mathcal{U}_{\mathrm{known}}$ is nonconvex one adds a projection term δ_{proj} as in Proposition E.24. □

Approximate optimality under value consistency

The previous bounds compare GDC to an ideal latent controller induced by the learned potential. To turn this into an optimality statement, one additional assumption is necessary: the data-induced potential must approximate the same constrained objective whose optimum we want to recover. The following theorem makes this conditional statement explicit.

Definition E.23 (Oracle constrained horizon objective)**.** *For a selected context $c = (o, g, \mathcal{C}_{\mathrm{known}}, H^\star)$ and data-supported region $U_c \subset \widehat{\mathcal{M}}^*$, let $\Gamma_H(z, c)$ be the set of known-feasible horizon-$H^\star$ continuations starting from latent point z. The oracle constrained horizon objective is a function $\Phi_{\mathrm{opt}}(\cdot, c) : U_c \to \mathbb{R}$ whose minimizers are the constrained optimal latent decisions on this region. In the deterministic case one may take*

$$\Phi_{\mathrm{opt}}(z, c) = \inf_{\tau \in \Gamma_H(z,c)} C_{\mathrm{opt}}(\tau; c), \tag{119}$$

where C_{opt} contains the task cost and all known and implicit feasibility costs. In the entropy-regularized case, the infimum is replaced by the soft minimum $-\beta_{\mathrm{opt}}^{-1} \log \int_{\Gamma_H(z,c)} \exp[-\beta_{\mathrm{opt}} C_{\mathrm{opt}}(\tau; c)]\, \mathrm{d}\nu_z(\tau)$. The theorem below applies after fixing either convention. Additive constants in Φ_{opt} are immaterial.

Proposition E.24 (Asymptotic optimality; sequence form of Theorem D.6)**.** *Fix a selected context $c = (o, g, \mathcal{C}_{\mathrm{known}}, H^\star)$ and a compact geodesically convex data-supported region $U_c \subset \widehat{\mathcal{M}}^*$. Let $\Phi_{\mathrm{opt}}(\cdot, c) : U_c \to \mathbb{R}$ be the oracle constrained horizon objective for the same task, horizon, and known-feasible scaffold, and assume it has a unique minimizer*

$$z^\star = \arg\min_{z \in U_c} \Phi_{\mathrm{opt}}(z, c).$$

For a sequence of learned GDC controllers indexed by n, write $\Phi_n(z, c)$ for the learned potential $\Phi_{\mathrm{GDC}}(z, c)$ and let $\hat{z}_n \in U_c$ be the latent point returned by online refinement. Suppose:

(i) ***Uniform value consistency up to normalization.*** *There are constants $a_n(c)$ and errors $\alpha_n \geq 0$ such that*

$$\sup_{z \in U_c} |\Phi_n(z, c) - \Phi_{\mathrm{opt}}(z, c) - a_n(c)| \leq \alpha_n. \tag{120}$$

(ii) ***Approximate online minimization.*** *The refinement step is η_n-optimal on U_c:*

$$\Phi_n(\hat{z}_n, c) \leq \inf_{z \in U_c} \Phi_n(z, c) + \eta_n. \tag{121}$$

(iii) ***Local curvature.*** *$\Phi_{\mathrm{opt}}(\cdot, c)$ is m-strongly geodesically convex on U_c:*

$$\Phi_{\mathrm{opt}}(z, c) \geq \Phi_{\mathrm{opt}}(z^\star, c) + \frac{m}{2} d_G(z, z^\star)^2 \qquad \textit{for all } z \in U_c. \tag{122}$$

Then the oracle suboptimality gap and latent distance satisfy

$$0 \leq \Phi_{\mathrm{opt}}(\hat{z}_n, c) - \Phi_{\mathrm{opt}}(z^\star, c) \leq 2\alpha_n + \eta_n, \tag{123}$$

and

$$d_G(\hat{z}_n, z^\star) \leq \sqrt{\frac{2(2\alpha_n + \eta_n)}{m}}. \tag{124}$$

If the first-action decoder is K_{D_ψ}-Lipschitz on U_c, its first-action reconstruction error is at most $\delta_{\mathrm{dec},n}$, and the final projection changes the decoded action by at most $\delta_{\mathrm{proj},n}$, then the executed action $u_{t,n}$ satisfies

$$\|u_{t,n} - u_t^\star\| \leq K_{D_\psi} \sqrt{\frac{2(2\alpha_n + \eta_n)}{m}} + \delta_{\mathrm{dec},n} + \delta_{\mathrm{proj},n}. \tag{125}$$

Consequently, if $\alpha_n \to 0$, $\eta_n \to 0$, $\delta_{\mathrm{dec},n} \to 0$, and $\delta_{\mathrm{proj},n} \to 0$, then

$$\Phi_{\mathrm{opt}}(\hat{z}_n, c) \to \Phi_{\mathrm{opt}}(z^\star, c), \qquad d_G(\hat{z}_n, z^\star) \to 0, \qquad \|u_{t,n} - u_t^\star\| \to 0.$$

Proof. Fix n and abbreviate $\widehat{\Phi} = \Phi_n(\cdot, c)$, $\Phi^\star = \Phi_{\mathrm{opt}}(\cdot, c)$, and $a = a_n(c)$. By the value-consistency bound (120), evaluated at $\hat{z}_n$,

$$\Phi^\star(\hat{z}_n) \leq \widehat{\Phi}(\hat{z}_n) - a + \alpha_n.$$

The approximate-minimization condition (121) gives

$$\widehat{\Phi}(\hat{z}_n) \leq \widehat{\Phi}(z^\star) + \eta_n,$$

since $z^\star \in U_c$. Applying (120) again, now at $z^\star$, yields

$$\widehat{\Phi}(z^\star) \leq \Phi^\star(z^\star) + a + \alpha_n.$$

Combining the three inequalities cancels the arbitrary normalization constant a and gives

$$\Phi^\star(\hat{z}_n) - \Phi^\star(z^\star) \leq 2\alpha_n + \eta_n.$$

The lower bound in (123) follows from optimality of $z^\star$. Strong geodesic convexity (122) then implies

$$\frac{m}{2} d_G(\hat{z}_n, z^\star)^2 \leq \Phi^\star(\hat{z}_n) - \Phi^\star(z^\star) \leq 2\alpha_n + \eta_n,$$

which proves (124).

Let $D_\psi(z, c)_0$ denote the first decoded action. If $u_t^\star$ is the oracle first action decoded from $z^\star$, Lipschitz continuity on U_c gives

$$\|D_\psi(\hat{z}_n, c)_0 - D_\psi(z^\star, c)_0\| \leq K_{D_\psi} d_G(\hat{z}_n, z^\star).$$

The decoder reconstruction error contributes at most $\delta_{\mathrm{dec},n}$, and the final projection contributes at most $\delta_{\mathrm{proj},n}$ by assumption. Substituting the distance bound gives (125). If the four error sequences vanish, the displayed convergence statements follow immediately. □

Corollary E.25 (Coverage and convergence rate)**.** *In the setting of Proposition E.24, let $\mathcal{Z}_n(c) \subset U_c$ be the finite set of latent anchors covered by the offline data for context c, and define its fill distance*

$$h_n = \sup_{z \in U_c} \inf_{\zeta \in \mathcal{Z}_n(c)} d_G(z, \zeta). \tag{126}$$

Suppose $\Phi_n(\cdot, c)$ and $\Phi_{\mathrm{opt}}(\cdot, c)$ are Lipschitz on U_c with constants L_n and L_{opt}, and that the learned potential has anchor value error

$$\sup_{\zeta \in \mathcal{Z}_n(c)} |\Phi_n(\zeta, c) - \Phi_{\mathrm{opt}}(\zeta, c) - a_n(c)| \le \varepsilon_n. \tag{127}$$

If the online refinement after K local optimization steps returns $\hat{z}_{n,K}$ with error η_K, then

$$\Phi_{\mathrm{opt}}(\hat{z}_{n,K}, c) - \Phi_{\mathrm{opt}}(z^\star, c) \le 2\varepsilon_n + 2(L_n + L_{\mathrm{opt}})h_n + \eta_K, \tag{128}$$

$$d_G(\hat{z}_{n,K}, z^\star) \le \sqrt{\frac{2\left[2\varepsilon_n + 2(L_n + L_{\mathrm{opt}})h_n + \eta_K\right]}{m}}, \tag{129}$$

$$\|u_{t,n,K} - u_t^\star\| \le K_{D_\psi} \sqrt{\frac{2\left[2\varepsilon_n + 2(L_n + L_{\mathrm{opt}})h_n + \eta_K\right]}{m}} + \delta_{\mathrm{dec},n} + \delta_{\mathrm{proj},K}. \tag{130}$$

Therefore, if $\sup_n L_n < \infty$, $\varepsilon_n = \mathcal{O}(r_n)$, $h_n = \mathcal{O}(s_n)$, and $\eta_K = \mathcal{O}(q_K)$, then the value gap is $\mathcal{O}(r_n + s_n + q_K)$ and the first-action error is

$$\mathcal{O}\left(\sqrt{r_n + s_n + q_K} + \delta_{\mathrm{dec},n} + \delta_{\mathrm{proj},K}\right).$$

In particular, quasi-uniform deterministic coverage of a d-dimensional region gives $h_n = \mathcal{O}(n^{-1/d})$. If anchors are sampled iid from a distribution whose density is bounded below on U_c, then $h_n = \mathcal{O}_p((\log n/n)^{1/d})$. With a linearly convergent local refinement $\eta_K \le C_\eta e^{-\kappa K}$, the value gap becomes

$$\mathcal{O}\left(r_n + s_n + e^{-\kappa K}\right).$$

Proof. For any $z \in U_c$, choose an anchor $\zeta_z \in \mathcal{Z}_n(c)$ with $d_G(z, \zeta_z) \le h_n$. The triangle inequality gives

$$\begin{aligned}
&|\Phi_n(z, c) - \Phi_{\mathrm{opt}}(z, c) - a_n(c)| \\
&\quad \le |\Phi_n(z, c) - \Phi_n(\zeta_z, c)| + |\Phi_n(\zeta_z, c) - \Phi_{\mathrm{opt}}(\zeta_z, c) - a_n(c)| \\
&\qquad + |\Phi_{\mathrm{opt}}(\zeta_z, c) - \Phi_{\mathrm{opt}}(z, c)| \\
&\quad \le (L_n + L_{\mathrm{opt}})h_n + \varepsilon_n.
\end{aligned}$$

Thus the uniform value-consistency error in (120) can be chosen as $\alpha_n = \varepsilon_n + (L_n + L_{\mathrm{opt}})h_n$. Substituting this value of α_n into (123), (124), and (125) yields (128)–(130). The big-O statements follow by replacing each error sequence with its rate.

For iid coverage, cover U_c by at most Cr^{-d} geodesic balls of radius $r/2$. If the sampling density is bounded below, each such ball has probability mass at least cr^d for small enough r. A union bound gives

$$\Pr[h_n > r] \le Cr^{-d} \exp(-ncr^d).$$

Choosing $r = A(\log n/n)^{1/d}$ with A large enough makes this probability vanish, proving $h_n = \mathcal{O}_p((\log n/n)^{1/d})$. □

Remark E.26 (What the optimality result does and does not claim)**.** *Proposition E.24 is stronger than the feasible-progress guarantee and requires a stronger premise. It does not say that arbitrary progress-certified demonstrations are automatically globally optimal. It says that when the learned data potential becomes uniformly value-consistent with the oracle constrained objective on the selected data-supported region, and the online refinement error also vanishes, the GDC decision approaches the corresponding constrained optimal decision on that region.*

E.8 Physics–data interpolation

Interpolated control potential under partial physics

Definition E.27 (Physics–data control potential)**.** *For $\lambda \in [0, 1]$, the interpolated control potential on $\widehat{\mathcal{M}}^*$ is*

$$\Phi_\lambda(z) = \lambda\, \Phi_{\mathrm{known}}(z) + (1 - \lambda)\, \Phi_{\mathrm{data}}(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star) + \Phi_{\mathrm{task}}(z). \tag{131}$$

INTERPOLATION THEOREM

Proposition E.28 (Physics–data interpolation under partial physics)**.** *Let $\lambda \in [0,1]$ and define the control law*

$$\dot{z} = -\operatorname{grad}_G \Phi_\lambda(z). \tag{132}$$

Then:

(i) At $\lambda = 1$, (132) *uses only the explicit known-physics potential and the task potential.*

(ii) At $\lambda = 0$, (132) *uses the data-driven progress potential $\Phi_{\mathrm{data}} + \Phi_{\mathrm{task}}$, where Φ_{data} supplies implicit progress and dynamics information from feasible trajectory data with progress certificates.*

(iii) For $\lambda \in (0,1)$, (132) *is a hybrid controller on $\widehat{\mathcal{M}}^*$ interpolating between explicit known-physics control and data-driven completion, with known-feasible support, anchor penalties, and singular perturbation structure.*

(iv) The Riemannian gradient has the explicit form:

$$\operatorname{grad}_G \Phi_\lambda = G^{-1}[\lambda \nabla_z \Phi_{\mathrm{known}} - (1-\lambda)\, s_\theta(z, o, g, \mathcal{C}_{\mathrm{known}}, H^\star) + \nabla_z \Phi_{\mathrm{task}}]\,. \tag{133}$$

Proof. The potential is affine in λ:

$$\Phi_\lambda = \lambda \Phi_{\mathrm{known}} + (1-\lambda)\Phi_{\mathrm{data}} + \Phi_{\mathrm{task}}.$$

Substituting $\lambda = 1$ removes the data term and yields $\Phi_{\mathrm{known}} + \Phi_{\mathrm{task}}$, proving (i). Substituting $\lambda = 0$ removes the known-potential weight from this interpolated term and yields $\Phi_{\mathrm{data}} + \Phi_{\mathrm{task}}$, proving (ii). For any $\lambda \in (0,1)$, both Φ_{known} and Φ_{data} enter the same scalar potential on the same manifold $\widehat{\mathcal{M}}^*$, while the support restriction, anchors, filters, and projection layer remain those of the known scaffold; hence the induced flow is the stated hybrid controller, proving (iii).

For (iv), differentiate:

$$\nabla_z \Phi_\lambda = \lambda \nabla_z \Phi_{\mathrm{known}} + (1-\lambda)\nabla_z \Phi_{\mathrm{data}} + \nabla_z \Phi_{\mathrm{task}}.$$

Since $\Phi_{\mathrm{data}} = -\log p_\theta^Z$, we have $\nabla_z \Phi_{\mathrm{data}} = -s_\theta$. Multiplying by G^{-1} gives the displayed Riemannian gradient formula. □

DATA-PHYSICS TRADEOFF PATH

Corollary E.29 (Data-physics tradeoff path on $\widehat{\mathcal{M}}^*$)**.** *Suppose the relevant data-supported region $K \subset \widehat{\mathcal{M}}^*$ is compact and each Φ_λ is continuous on K. Then minimizers of $\Phi_\lambda(z)$ on K exist for every $\lambda \in [0,1]$. If the minimizer is unique for each λ, the minima trace a continuous data-physics tradeoff path*

$$\mathcal{P} = \left\{ z^\star(\lambda) = \operatorname*{arg\,min}_{z \in K} \Phi_\lambda(z) \;\middle|\; \lambda \in [0,1] \right\}. \tag{134}$$

Proof. Existence follows from the Weierstrass theorem because Φ_λ is continuous on the compact region K. The joint map $(z, \lambda) \mapsto \Phi_\lambda(z)$ is continuous since it is affine in λ and continuous in z. Berge's maximum theorem implies that the argmin correspondence is nonempty, compact-valued, and upper hemicontinuous. When the minimizer is unique for every λ, this correspondence is single-valued; a single-valued upper hemicontinuous argmin map on a compact parameter interval is continuous. Therefore $z^\star(\lambda)$ traces the stated continuous tradeoff path. □

E.9 ONLINE CONTROL USED IN THE THEORY

The stability and interpolation results above analyze the online controller defined in Appendix C.3. In particular, the learned models are frozen, the controller operates in latent space, known constraints are checked after decoding through D_ψ, and only the first action of the selected segment is executed.

E.10 Wasserstein gradient flow interpretation

GDC as Wasserstein gradient flow on the empirical manifold

Proposition E.30 (Wasserstein gradient-flow form; cf. Jordan et al., 1998)**.** *Under Assumption E.1, the Fokker–Planck evolution* (103) *on $\widehat{\mathcal{M}}^*$ is the Wasserstein gradient flow of the free energy:*

$$\mathcal{F}[p] = \mathbb{E}_p\big[-\log p_\theta^Z(z \mid o, g, \mathcal{C}_{\text{known}}, H^\star) + \Phi_{\text{known}}(z) + \Phi_{\text{task}}(z)\big] - \beta_{\text{L}}^{-1} H_G(p), \tag{135}$$

where the Wasserstein metric is computed with respect to geodesic distances on $(\widehat{\mathcal{M}}^*, G)$.

Proof. Write

$$\mathcal{F}[p] = \int_{\widehat{\mathcal{M}}^*} p(z)\Phi_{\text{GDC}}(z)\,\text{dvol}_G + \beta_{\text{L}}^{-1}\int_{\widehat{\mathcal{M}}^*} p(z)\log p(z)\,\text{dvol}_G.$$

Its first variation is

$$\frac{\delta\mathcal{F}}{\delta p} = \Phi_{\text{GDC}} + \beta_{\text{L}}^{-1}(\log p + 1).$$

In the 2-Wasserstein geometry induced by the Riemannian metric G, the gradient-flow equation for $\mathcal{F}$ is the continuity equation

$$\partial_t p = \text{div}_G\left(p\ \text{grad}_G\,\frac{\delta\mathcal{F}}{\delta p}\right).$$

Dropping the additive constant in the first variation gives

$$\partial_t p = \text{div}_G(p\ \text{grad}_G\,\Phi_{\text{GDC}}) + \beta_{\text{L}}^{-1}\,\text{div}_G(p\ \text{grad}_G\log p)\,.$$

Since $p\ \text{grad}_G\log p = \text{grad}_G\, p$, the second term is $\beta_{\text{L}}^{-1}\Delta_G p$. This is exactly (103). □

In this frozen-context idealization, the stochastic variant is a steepest-descent flow in probability space over $\widehat{\mathcal{M}}^*$ that trades off the expected control potential (known physics, data-induced implicit information, and task cost) against behavioral entropy on the manifold; the reported experiments use the deterministic controller.

F Full Experimental Protocols and Results

We instantiate GDC in two domains with different meanings of partial structural knowledge. In SUMO, the known structure consists of explicit traffic and actuator constraints, while route-directed behavior is learned from weak driving traces. In the bilevel problem, the known structure consists of the nested problem form, analytic objectives and gradients, feasibility projection, and local relaxation updates, while useful joint motion across the levels is learned from solver traces. Each domain is evaluated on its own numerical benchmark: the multilevel problems test whether the learned progress field supplies fast cross-level directions beyond a known local scaffold, and the SUMO routes test whether the same principle improves closed-loop route progress while explicit traffic constraints remain enforced online.

F.1 Application I: Route-progress control in SUMO

Task and known scaffold

SUMO gives a concrete partial-physics instance of GDC (Lopez et al., 2018; Krajzewicz et al., 2012). The ego vehicle must complete a sampled route through a traffic network, but the controller is not treated as an end-to-end perception system. SUMO provides localization, lane, leader-vehicle, and traffic-light signals; explicit rules check speed, acceleration, headway, lane-change, and collision constraints. The missing component is the local direction of route progress: among actions that satisfy these known tests, the controller must decide which one advances the route without relying on a full online optimal-control solve.

At each control step the controller observes

$$o_t = (p_t^x, p_t^y, v_t, \sin\psi_t, \cos\psi_t, \ell_t, s_t, L_t, v_t^{\max}, \iota_t, N_t^{\text{route}}, d_t, g_t^{\text{lead}}, \Delta v_t^{\text{lead}}, c_t^{\text{tls}}, c_t^{\text{col}}, c_t^{\text{arr}}) \in \mathbb{R}^{17}, \tag{136}$$

where d_t is the remaining route distance. The action $u_t = (a_t, q_t)$ contains a longitudinal acceleration and a lane-change command $q_t \in \{-1, 0, 1\}$. After executing one action in SUMO, the controller receives a new observation and replans.

The known-feasible filter checks

$$-a_{\text{dec}}^{\max} \le a_t \le a_{\text{acc}}^{\max}, \qquad 0 \le v_t + \Delta t_{\text{env}}\, a_t \le v_t^{\max} + \varepsilon_v, \tag{137}$$

$$q_t \in \{-1, 0, 1\}, \qquad |q_t - q_{t-1}| \le \Delta q_{\max}, \tag{138}$$

$$g_t^{\text{lead}} \ge g_{\min} + T_h v_t, \qquad c_t^{\text{col}} = 0. \tag{139}$$

We use $a_{\text{acc}}^{\max} = 2.0\,\text{m/s}^2$, $a_{\text{dec}}^{\max} = 4.5\,\text{m/s}^2$, $g_{\min} = 2.5\,\text{m}$, $T_h = 0.8\,\text{s}$, and environment control period $\Delta t_{\text{env}} = 0.5\,\text{s}$ in simulated time (distinct from the latent integration step Δt of the online navigator). This simulator step is not a wall-clock computation deadline. These constraints determine what may be executed, but they do not select among feasible route choices. A conservative fallback segment is constructed from this same scaffold by tracking a target speed, braking for near leaders or restrictive signals, and avoiding discretionary lane changes. The fallback is a feasible anchor and a last-resort action source; it is not the learned policy.

Progress data and online decision rule

Offline rollouts come from a deliberately weak and diverse behavior policy with randomized acceleration, braking, coasting, exploration, and lane changes. It observes traffic-light state and slows at red or yellow signals, but is neither expert nor globally optimal. The data therefore include progress, waiting, braking, and imperfect local maneuvers. GDC does not treat these traces as expert labels. It uses them to learn which feasible segments tend to reduce remaining route distance.

Each rollout is divided into action segments

$$\tau_t = (u_t, u_{t+1}, \ldots, u_{t+H-1}). \tag{140}$$

Each segment receives the signed certificate

$$\Delta\rho_{\text{SUMO}}(\tau_t, g) = d_t - d_{t+H} - \kappa_{\text{col}} N_{\text{col}}(\tau_t) + \kappa_{\text{arr}} \mathbb{I}\{\text{arrival in } \tau_t\}. \tag{141}$$

The target distribution is the progress-tilted empirical distribution

$$q_\beta^{\text{SUMO}}(\tau \mid o, g, \mathcal{C}_{\text{known}}) \propto q_{\mathcal{D}}^{\text{SUMO}}(\tau \mid o, g, \mathcal{C}_{\text{known}}) \exp(\beta\ \text{clip}(\Delta\rho_{\text{SUMO}}(\tau, g), -c, c)). \tag{142}$$

At test time, GDC samples or initializes candidate segments near the fallback and near data-supported local motions, refines them with the learned score, decodes $\tau_t(z) = D_\psi(z, o_t, g, \mathcal{C}_{\text{known}}, H_t^\star)$, filters them with (137)–(139), and ranks the remaining candidates by

$$\begin{aligned} J_t(z) = {} & w_E \Phi_{\text{data}}(z, o_t, g, \mathcal{C}_{\text{known}}, H_t^\star) - w_P \widehat{\Delta\rho}_{\text{SUMO}}(\tau_t(z), g) \\ & + w_U \|\tau_t(z)\|^2 + w_C V_{\mathcal{C}_{\text{known}}}(\tau_t(z)) + w_T \|z - z_t^{\text{fb}}\|^2. \end{aligned} \tag{143}$$

Here $V_{\mathcal{C}_{\text{known}}}$ is the aggregate violation returned by the explicit known-constraint checker, and z_t^{fb} is the latent representation of the fallback segment. Only the first action of the selected segment is executed. The next observation starts a new planning step. Thus the learned distribution is not executed open loop; it supplies a progress energy for choosing among known-feasible short motions. If no learned candidate improves sufficiently over the fallback, the controller executes the first fallback action.

Numerical results

The SUMO evaluation tests the same role separation in a closed-loop driving task: explicit traffic constraints decide what can be executed, while the learned progress energy chooses among feasible local motions. We train from 100,000 diverse weak rollouts and retain at most five million selected segments per horizon. Online evaluation uses 100 closed-loop episodes. At every control step the controller scores candidate segments, filters them with the known scaffold, executes only the first action, and replans after the next SUMO observation.

Benchmarks. We use three increasingly difficult route-progress benchmarks (Table 3), all based on perturbed 20×20 two-lane grids with $120\,\mathrm{m}$ nominal edge length and $13.89\,\mathrm{m/s}$ nominal speed. The priority benchmark is the clean route-completion case. Sparse TLS adds traffic-light-signal (TLS) delay, where stopping may be the correct local action. Complex TLS is the harder-map benchmark: it keeps the graph connected but adds 18 sparse traffic lights, 64 very-low-speed detour edges at $2.5\,\mathrm{m/s}$, and 76 bottleneck edges at $7.5\,\mathrm{m/s}$ on a grid perturbed at a $14\,\mathrm{m}$ scale. This creates routes where local progress, stopping, and longer-range route intent can conflict.

Table 3: SUMO route-progress benchmark suite. All maps are perturbed 20×20 two-lane grids. Multi-H denotes joint online selection over $H \in \{6, 12, 24\}$; fixed-H restricts candidate selection to $H = 12$.

Benchmark	TLS	Slow detour	Bottleneck	Removed edges	Controllers
Priority grid	0	0	0	0	fixed $H = 12$
Sparse TLS	sparse	0	0	0	fixed $H = 12$, multi-H
Complex TLS	18	64 at $2.5\,\mathrm{m/s}$	76 at $7.5\,\mathrm{m/s}$	0	fixed $H = 12$, multi-H

Controllers and metrics. The fixed-horizon baseline and the multi-horizon controller use the same observation features, weak rollout source, known-feasible filter, fallback construction, and receding-horizon first-action execution. The difference is only the online temporal support: fixed-H scores $H = 12$ segments, while multi-H scores candidates from $H \in \{6, 12, 24\}$ in a single pool and executes the first action of the best safe segment. We report arrival rate, normalized route progress, collisions, mean and maximum known-constraint violation, non-fallback fraction, mean speed, and episode length.

Table 4: SUMO closed-loop results over 100 episodes. Arr. is arrival rate, Prog. is normalized route progress, Coll. is mean collision count, Non-fb. is the fraction of steps where a learned candidate rather than the fallback supplies the executed action, Speed is mean speed in $\mathrm{m/s}$, and Steps is mean episode length.

Benchmark	Controller	Arr.	Prog.	Coll.	Mean viol.	Max viol.	Non-fb.	Speed	Steps
Priority grid	fixed $H = 12$	1.000	1.000	0.000	1.22×10^{-6}	2.80×10^{-4}	0.999	11.24	202.6
Sparse TLS	fixed $H = 12$	1.000	1.000	0.000	1.48×10^{-7}	5.01×10^{-5}	1.000	10.97	230.3
Sparse TLS	multi-H	1.000	1.000	0.000	0.00	0.00	0.997	10.58	233.9
Complex TLS	fixed $H = 12$	1.000	1.000	0.000	1.59×10^{-5}	5.88×10^{-3}	0.999	9.25	317.3
Complex TLS	multi-H	1.000	1.000	0.000	0.00	0.00	0.991	8.96	322.9

Results. The GDC configurations in Table 4 reach every route goal with zero collisions. The priority result is the clean route-completion case: GDC achieves unit progress with only numerical-tolerance known-constraint violation. Both fixed-H and multi-H complete all reported routes. The harder complex map is where the horizon comparison becomes meaningful: multi-H has zero reported residual violation, and the two settings select different candidates while sharing the same known-feasible execution layer. Table 4 reports speed and episode length alongside arrival and feasibility metrics.

Figure 5 gives the qualitative explanation for the complex-map result. Multi-H does not vote among three independent policies. It scores all safe candidate segments together and executes the first action of the best segment. The selected horizon therefore acts as an online diagnostic of temporal scale: longer horizons dominate during ordinary progress, while shorter horizons appear near local traffic or constraint-sensitive events. The batched implementation gives the same aggregate statistics as the unbatched multi-horizon controller, showing that this candidate-scoring step can be batched without changing the closed-loop behavior.

Comparison with rule, imitation, and RL baselines. Table 5 compares GDC with a rule safety controller, the weak behavior source used for data collection, behavior cloning on the offline rollouts, and PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018) baselines. The comparison uses the same sparse-TLS and complex-TLS route suites. GDC is the only method that reaches 100%

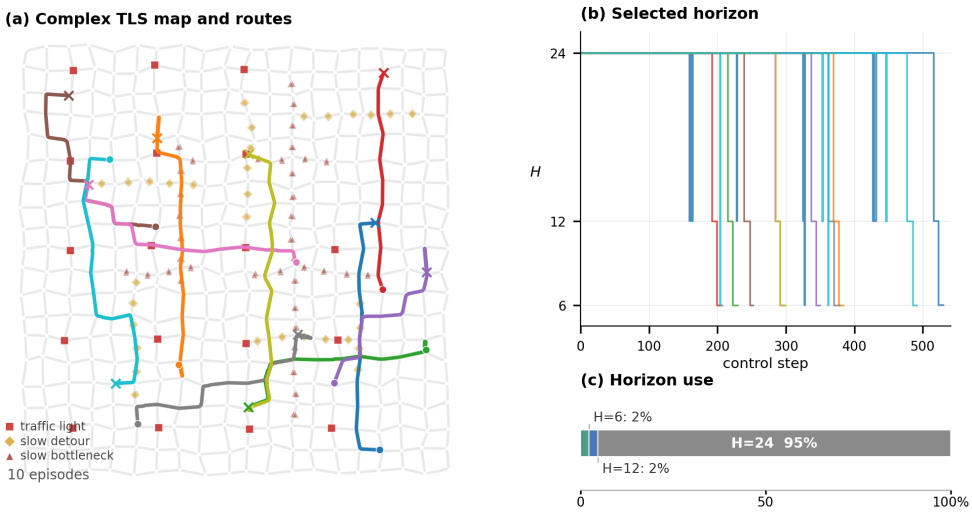


Figure 5: Complex-map multi-horizon diagnostic over ten representative episodes. (a) Routes through lights, detours, and bottlenecks. (b) Selected horizon over control steps. (c) Aggregate use: $H = 24$ for 95% of decisions and shorter support near local events. The batched candidate-scoring evaluation also reports the horizon-use distribution over all 100 episodes on this map with sparse traffic lights, slow bottleneck corridors, and slow detour edges.

arrival on both maps with zero collisions and zero mean known-constraint violation in the multi-horizon setting. On the harder complex TLS benchmark, GDC multi-H improves arrival from 91% for the rule controller, 51% for behavior cloning, 81% for PPO, and 79% for SAC to 100%, while also reducing the mean episode length relative to all non-GDC baselines.

Table 5: SUMO controller benchmark comparison. GDC and non-SAC rows are evaluated over 100 episodes; the available sparse-TLS SAC run contains 90 episodes. Prog. is normalized route progress, Arr. is arrival rate, Coll. is mean collision count, Speed is mean speed in $\mathrm{m/s}$, and Steps is mean episode length.

Benchmark	Method	Prog.	Arr.	Coll.	Mean viol.	Speed	Steps
Sparse TLS	GDC fixed-H12	1.000	1.000	0.00	1.48×10^{-7}	10.97	230.3
Sparse TLS	GDC multi-H	**1.000**	**1.000**	0.00	**0.00**	10.58	**233.9**
Sparse TLS	Rule safety	1.000	0.990	0.00	1.33×10^{-1}	11.05	330.6
Sparse TLS	Weak behavior	0.962	0.840	0.00	1.44×10^{-2}	7.86	449.8
Sparse TLS	Behavior cloning	0.931	0.760	0.00	3.55×10^{-3}	6.26	537.1
Sparse TLS	PPO	0.997	0.970	0.00	1.27×10^{-2}	7.87	451.4
Sparse TLS	SAC	1.000	1.000	0.00	1.82×10^{-2}	10.72	346.3
Complex TLS	GDC fixed-H12	1.000	1.000	0.00	1.59×10^{-5}	**9.25**	**317.3**
Complex TLS	GDC multi-H	**1.000**	**1.000**	0.00	**0.00**	8.96	322.9
Complex TLS	Rule safety	0.989	0.910	0.00	1.48×10^{-1}	9.00	441.2
Complex TLS	Weak behavior	0.841	0.550	0.00	1.44×10^{-2}	5.65	546.9
Complex TLS	Behavior cloning	0.770	0.510	0.00	3.90×10^{-3}	4.74	587.7
Complex TLS	PPO	0.965	0.810	0.00	1.29×10^{-2}	8.36	461.7
Complex TLS	SAC	0.960	0.790	0.00	2.65×10^{-2}	8.51	436.0

F.2 Additional SUMO Diagnostics

Figure 6 provides an additional diagnostic for the larger 60×60 complex TLS map. On the two 60×60 maps (Figures 2 and 6) the multi-horizon candidate pool is $\mathcal{H} = \{6, 24, 48\}$, while the 20×20 maps and the transfer maps (Figures 3, 7, and 8) use $\mathcal{H} = \{6, 12, 24\}$. The long-route stress test is promoted to Figure 2. Table 6 attaches each SUMO diagnostic figure to the numerical

conclusion it supports, including the two held-out transfer figures below. Traffic-light constraints are rendered as signal markers. Star markers are reserved for slow or other non-TLS roadblocks; none are present in this TLS-only scalability map.

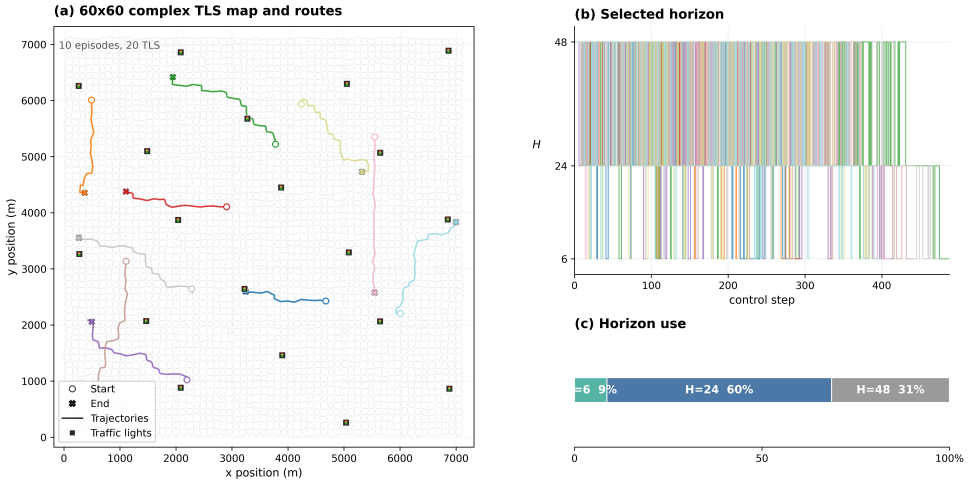


Figure 6: Complex TLS scalability diagnostic on the 60×60 map. The figure shows ten recorded ego paths, the selected planning horizon at each control step, and the aggregate horizon-use distribution for this scenario.

Table 6: Summary of the SUMO diagnostic figures. Bracketed route lengths report minimum and maximum values. Transfer I fixes the base topology and changes disturbances, while Transfer II changes graph parameters and disturbances together. The Transfer I row summarizes a separate random draw of held-out routes rather than the routes plotted in Figure 7; the Transfer II row summarizes the traces shown in Figure 8. The 60×60 rows use the candidate pool $\mathcal{H} = \{6, 24, 48\}$; all other rows use $\mathcal{H} = \{6, 12, 24\}$.

Diagnostic	Routes / maps	Arr./Prog.	Coll.	Steps	Stressor / diagnostic signal
Main complex TLS, Fig. 5	100 eps.; 1.15 [0.85, 1.44] km.	1.000/1.000	0	322.9	H=24 on 95.3% of decisions; mean speed 8.96 m/s.
60×60 TLS, Fig. 6	10 routes; 1.72 [1.20, 2.19] km.	1.000/1.000	0	361.1	H=24/48 on 91.4% of decisions; mean speed 12.41 m/s.
60×60 long TLS, Fig. 2	10 routes; 7.43 [7.03, 8.09] km.	1.000/1.000	0	1578.0	7.0–8.1 km routes; H=24/48 on 75.2% of decisions.
Transfer I: blocked map, Fig. 7	4 maps, 20 routes; 1.97 [1.49, 2.78] km.	1.000/1.000	0	252.0	Fixed topology; held-out closures and slow roads; 100-map benchmark: zero known violation, 99.7% non-fallback.
Transfer II: graph + disturbance, Fig. 8	4 shown maps, 20 traces; 2.32 [1.90, 3.06] km.	1.000/1.000	0	579.9	Unseen graph parameters plus disturbances: 18×19–22×22 grids, 8–26 closure pairs, 33–48 bottleneck pairs.

SUMO Transfer Diagnostics

We also include representative qualitative transfer diagnostics for the two held-out SUMO transfer settings. *Transfer I: blocked-map transfer* keeps the base perturbed road topology fixed and changes the deployment map through held-out road closures and slow/bottleneck edges. This tests whether GDC can reuse the learned progress direction when the feasible scaffold is locally cut or slowed, rather than only replaying routes from the training map. *Transfer II: graph-and-disturbance transfer* changes the generated graph itself, including grid dimensions, lane count, edge length, speed, traffic-light spacing, and geometric perturbation, while also introducing new closures and bottlenecks. This harder setting tests whether the same progress model remains useful when both the scaffold geometry and the local disturbances move out of the original graph. In both settings, the online controller receives the current known map constraints as $\mathcal{C}_{\text{known}}$; transfer therefore concerns the learned direction field on top of a new scaffold, not blind map inference. We show representative maps rather than the full set of ten visualized maps per setting.

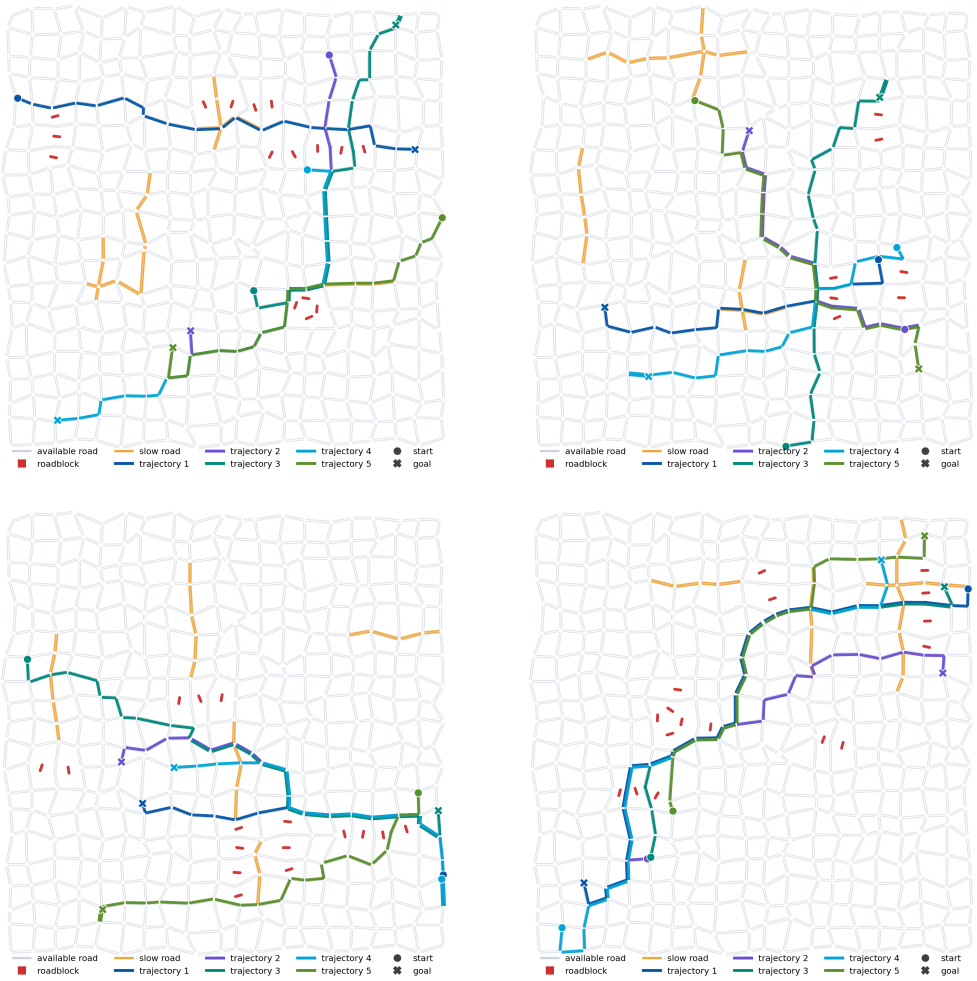


Figure 7: Representative blocked-map transfer diagnostics. Four held-out maps are shown; each panel overlays five long routes. Red marks closures, orange marks slow/bottleneck roads, and the 1.7–2.4 km routes make transfer-time detours inspectable.

Blocked-map transfer. This setting holds the base perturbed topology fixed and changes only held-out closures and slow roads. On the corresponding 100-map benchmark, GDC keeps arrival/progress at $1.000/1.000$, has zero known violations, and uses non-fallback learned control on 99.7% of decisions; the plotted detours therefore summarize the same transfer behavior rather than only qualitative overlays.

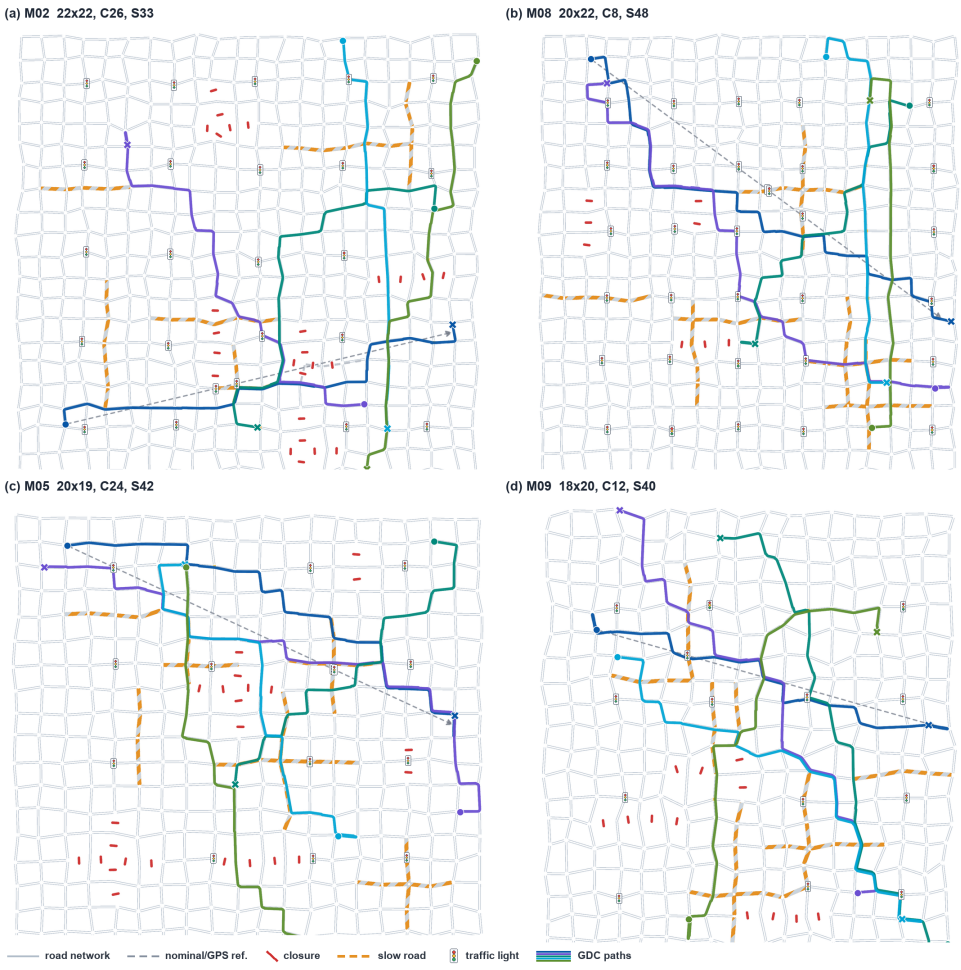


Figure 8: Held-out graph-and-disturbance transfer examples. Panels use unseen graph parameters and disturbances; red ticks mark closures, dashed orange marks slow roads, icons mark traffic lights, and colored curves are successful long GDC routes.

Graph-and-disturbance transfer. This harder setting changes both the generated graph and its local disturbances. The shown maps span 18×19 to 22×22 grids, 8–26 closure pairs, and 33–48 bottleneck pairs; the 20 shown closed-loop traces still reach their route goals with zero collisions.

SUMO SEED SENSITIVITY

Table 7 separates test-route randomness from checkpoint-training randomness. The frozen-checkpoint comparison uses the main test batch and additional held-out test batches; the former has 100 episodes and each additional batch has 25. The independent training check fits $H = 12$ models with the same segment data, architecture, and 80-epoch budget, then evaluates them on common 25-episode routes. Means and standard deviations below are over run means, not over pooled episodes. Across the test batches, multi-H preserves arrival and zero recorded known-constraint residuals.

Table 7: SUMO seed-sensitivity checks. Values are mean ± standard deviation across run means. No collisions were recorded in these runs. The training-seed row evaluates independently fitted $H = 12$ checkpoints on the same test routes; it is not a multi-H retraining result.

Variation source	Controller	Arrival	Steps	Speed (m/s)	Mean max. violation
Test seed, frozen checkpoint	Fixed $H = 12$	0.99 ± 0.02	298.6 ± 23.0	9.52 ± 0.55	0.00968 ± 0.01542
Test seed, frozen checkpoint	Multi-H	1.00 ± 0.00	318.9 ± 4.2	8.91 ± 0.17	0
Independent training seed	Fixed $H = 12$	0.97 ± 0.038	334.9 ± 19.2	9.01 ± 0.25	0.00069 ± 0.00138

F.3 Application II: Hierarchically coupled merit optimization

Problem family and known structural knowledge

For each parameter vector ϑ, consider a nonconvex parameterized two-level objective family over $y, x \in \mathbb{R}^d$ (Colson et al., 2007; Dempe & Zemkoho, 2020). The lower-level merit is a quartic-well objective

$$f_{\mathrm{lower},\vartheta}(x; y) = \frac{1}{4d} \|x \odot x - \ell_\vartheta(y)\|_2^2 + \frac{\mu_x}{2d} \|x\|_2^2 + \frac{\lambda_x}{2(d-1)} \sum_{i=1}^{d-1} (x_{i+1} - x_i)^2, \qquad \ell_\vartheta(y) = a_\vartheta + M_\vartheta y, \tag{144}$$

where $M_\vartheta = M_0 + \sum_{r=1}^{R} s_r u_r v_r^\top$ is a low-rank perturbation of a fixed banded coupling. The upper objective is also nonconvex:

$$F_{\mathrm{upper},\vartheta}(y, x) = \frac{\alpha_y}{d} \|y - g_y\|_2^2 + \frac{\alpha_x}{d} \|x - g_x\|_2^2 + \frac{\kappa_y}{d} \|y \odot y - r_y\|_2^2 + \frac{\kappa_x}{d} \|x \odot x - r_x\|_2^2 + \frac{\eta_{xy}}{d} \|x - y\|_2^2 + \frac{1}{d} x^\top C y + \frac{\lambda_y}{d-1} \sum_{i=1}^{d-1} (y_{i+1} - y_i)^2. \tag{145}$$

The parameter vector ϑ contains the goals (g_y, g_x), lower shifts a_ϑ, well radii (r_y, r_x), low-rank scales $(s_r)_{r=1}^R$, and the ridge parameter μ_x. The quartic terms make both the lower and upper landscapes nonconvex, with multiple attractive basins. The corresponding exact bilevel target is

$$\min_{y \in [-2,2]^d} F_{\mathrm{upper},\vartheta}\big(y, x_\vartheta^\star(y)\big), \qquad x_\vartheta^\star(y) \in \operatorname*{arg\,min}_{x \in [-2,2]^d} f_{\mathrm{lower},\vartheta}(x; y). \tag{146}$$

The reported online and SCIP experiments below evaluate a finite-weight joint merit rather than enforcing $x = x_\vartheta^\star(y)$. The known structural knowledge consists of the level decomposition, analytic objectives, gradients, box projection, step-size projection, and local lower-level update; it does not include a globally certified lower response or upper optimum.

The action is a joint increment $u = (\Delta y, \Delta x)$. The explicit scaffold enforces

$$y^+, x^+ \in [-2, 2]^d, \qquad \|\Delta y\|_2 \le \delta_y, \qquad \|\Delta x\|_2 \le \delta_x. \tag{147}$$

The trilevel scaling experiment uses the same nonconvex construction with state $\omega = (z, y, x)$. The middle and lower objectives are quartic wells around parameterized targets $m_\vartheta(z)$ and $\ell_\vartheta(y, z)$:

$$f_{\mathrm{middle},\vartheta}(y; z) = \frac{1}{4d} \|y \odot y - m_\vartheta(z)\|_2^2 + \frac{\mu_y}{2d} \|y\|_2^2 + \frac{\lambda_y}{2(d-1)} \sum_{i=1}^{d-1} (y_{i+1} - y_i)^2,$$

$$f_{\mathrm{lower},\vartheta}(x; y, z) = \frac{1}{4d} \|x \odot x - \ell_\vartheta(y, z)\|_2^2 + \frac{\mu_x}{2d} \|x\|_2^2 + \frac{\lambda_x}{2(d-1)} \sum_{i=1}^{d-1} (x_{i+1} - x_i)^2. \tag{148}$$

The top objective adds quadratic goal terms, quartic well terms for z, y, x, pairwise cross-level penalties, and indefinite bilinear couplings. The corresponding exact trilevel target is

$$\begin{aligned} \min_{z \in [-2,2]^d} \quad & F_{\mathrm{top},\vartheta}\big(z, y_\vartheta^\star(z), x_\vartheta^\star(z, y_\vartheta^\star(z))\big), \\ & y_\vartheta^\star(z) \in \operatorname*{arg\,min}_{y \in [-2,2]^d} f_{\mathrm{middle},\vartheta}(y; z), \\ & x_\vartheta^\star(z, y) \in \operatorname*{arg\,min}_{x \in [-2,2]^d} f_{\mathrm{lower},\vartheta}(x; y, z). \end{aligned} \tag{149}$$

The top level depends on (z, y, x), the middle level depends on $(y; z)$, and the lower level depends on $(x; y, z)$. The reported trilevel-family experiment uses the finite-weight joint merit below. Box and step projections are applied separately to z, y, and x.

Progress model and online solver

Weak solver traces are divided into segments of $H = 8$ joint increments and conditioned on $(\omega_t, u_{t-1}, \vartheta)$. Segment labels use decrease in a fixed-weight scalar progress score minus effort:

$$\Delta\rho_{\mathrm{bi}}(\tau_t, \vartheta) = \Phi^{\mathrm{bi}}_{\vartheta}(\omega_t) - \Phi^{\mathrm{bi}}_{\vartheta}(\omega_{t+H}) - \eta \sum_{k=0}^{H-1} \|u_{t+k}\|^2. \tag{150}$$

This scalar score is a training and selection certificate for candidate motion. It also defines the single-level penalized joint program used for the online and offline-reference comparisons; exact nested response constraints in (146) and (149) are not enforced. The training distribution is tilted by $\exp(\beta\Delta\rho_{\mathrm{bi}}(\tau, \vartheta))$. Online, GDC generates joint segments, refines them using the learned score, projects their increments into (147), and compares them with a known-only local update. Selection combines learned energy, predicted decrease in the scalar score, lower merit, and a trust-region penalty. Only the first increment is applied; feasibility or progress failure triggers the known-only fallback.

Protocol and metrics

Training uses $50{,}000$ parameterized traces, with goals, well locations, low-rank couplings, offsets, and ridge parameters resampled per trace. We compare GDC with known-only and nested local online solvers on held-out parameters. For evaluation only, offline multistart search and SCIP branch-and-cut solves of the same box-constrained penalized joint programs provide reference values or incumbents; they are not online runtime competitors. The numerical scaling tables report per-instance results across dimensions. We report

$$\mathrm{ngap} = \frac{\Phi_{\mathrm{GDC}} - \Phi_{\mathrm{ref}}}{\Phi_{\mathrm{init}} - \Phi_{\mathrm{ref}}}. \tag{151}$$

All reported iterates satisfy the explicit box, step, and known-feasibility constraints. The lower levels enter through their objectives, local relaxation updates, and merit terms in candidate selection. The scalar gap measures proximity to a local multistart reference under Φ; it does not certify a global best response at each lower level. This experiment tests whether one learned progress field provides useful cross-level directions on unseen nonconvex instances while the explicit scaffold maintains the known box and step constraints.

Numerical results

The multilevel evaluation treats GDC as an amortized online controller for parameterized bilevel and trilevel nonconvex optimization families. The bilevel experiment uses the family defined above. The trilevel experiment extends the same test from a state $\omega = (y, x)$ to $\omega = (z, y, x)$ with top, middle, and lower variables. In both settings, each instance has its own parameter vector, and the controller receives analytic objectives, gradients, local relaxation updates, and known feasibility projection, but not a globally certified solution. The role of GDC is therefore to learn useful joint motion across levels from weak solver traces, while the known scaffold keeps the online updates inside the explicit box and step constraints.

For the tables and figures, we evaluate the final iterates using fixed-weight scalar criteria. The online methods are compared on these criteria, which also define the penalized joint programs solved by the offline references and SCIP; the nested response conditions in (146) and (149) are penalized rather than enforced. For bilevel instances the reported evaluation criterion is

$$\Phi(y, x) = F_{\mathrm{upper}}(y, x) + \lambda_{\mathrm{lower}}\, f_{\mathrm{lower}}(x; y), \tag{152}$$

and for trilevel instances the reported evaluation criterion is

$$\Phi(z, y, x) = F_{\mathrm{top}}(z, y, x) + \lambda_{\mathrm{middle}}\, f_{\mathrm{middle}}(y; z) + \lambda_{\mathrm{lower}}\, f_{\mathrm{lower}}(x; y, z). \tag{153}$$

Tables 8 and 9 report held-out per-instance scaling results under this evaluation criterion. The normalized GDC gap is $(\Phi_{\mathrm{GDC}} - \Phi_{\mathrm{ref}})/(\Phi_{\mathrm{init}} - \Phi_{\mathrm{ref}})$. The reference value is an offline multistart search under the same evaluation criterion; it is used only as an evaluation reference, not as a fair online runtime competitor.

Table 8: Bilevel evaluation-score scaling. GDC is the learned online controller; known and nested are conventional online baselines; the reference column is an offline evaluation reference under the same scalar criterion. Normalized gap is reported as $100(\Phi_{\mathrm{GDC}} - \Phi_{\mathrm{ref}})/(\Phi_{\mathrm{init}} - \Phi_{\mathrm{ref}})$; lower is better.

d	Ref. Φ (offline reference)	Known Φ	Nested Φ	GDC Φ	Gap (%)	GDC time (s)
6	0.122821439	0.727128	0.699750	**0.124122773**	**0.096**	44.659
8	0.085741765	0.620790	0.601718	**0.088084970**	**0.194**	57.622
10	0.118656182	0.542130	0.508923	**0.121497257**	**0.229**	52.206
12	0.132315706	1.005490	0.914664	**0.134485621**	**0.179**	83.117

Table 9: Trilevel evaluation-score scaling. Normalized gap is reported as a percentage; all four trilevel cases are below 0.65%.

d	Ref. Φ (offline reference)	Known Φ	Nested Φ	GDC Φ	Gap (%)	GDC time (s)
4	0.164844401	0.572960	0.658353	**0.168670984**	**0.345**	79.898
6	0.178681018	1.020626	1.144793	**0.180060025**	**0.093**	116.208
8	0.122788129	0.951728	1.072284	**0.131214128**	**0.640**	138.865
10	0.178549407	1.059038	1.146033	**0.181326304**	**0.187**	138.403

Table 10 reports the separate SCIP benchmark on the same penalized joint merit programs with a four-hour time limit per case. SCIP is used here as an offline branch-and-cut benchmark. We therefore report its incumbent evaluation score and optimality gap, and mark all time-limit cases as not globally certified.

Table 10: SCIP branch-and-cut benchmark on the penalized joint merit formulations with a four-hour time limit per case. Only rows with status "optimal" are globally certified. Rows with status "timelimit" report the best incumbent found within the limit.

Problem	d	Status	Gap	Incumbent Φ	Time (s)
Bilevel	6	optimal	0.000000	0.122821621	22.6
Bilevel	8	timelimit	0.000045	0.085741916	14401.7
Bilevel	10	timelimit	0.000060	0.118656277	14402.9
Bilevel	12	timelimit	0.000270	0.132315706	14400.0
Trilevel	4	optimal	0.000000	0.164844566	327.8
Trilevel	6	timelimit	0.000065	0.178681118	14403.2
Trilevel	8	timelimit	0.001114	0.122788129	14400.0
Trilevel	10	timelimit	0.090036	0.178549407	14400.0

Figure 9 compares online GDC wall time with SCIP branch-and-cut wall time after normalizing both by the four-hour SCIP budget; time-limit rows are treated as incumbent-only offline references. Figure 4 in the main text shows the GDC online trajectories for all reported bilevel and trilevel dimensions approaching their corresponding SCIP incumbents.

Across both multilevel families, GDC reaches evaluation scores very close to the offline reference search and SCIP incumbents. The bilevel normalized gaps are all below 0.23%, and the trilevel normalized gaps are all below 0.65%, the largest being 0.64% at $d = 8$. Known structure maintains the explicit box and step constraints, while GDC supplies a learned cross-level direction that

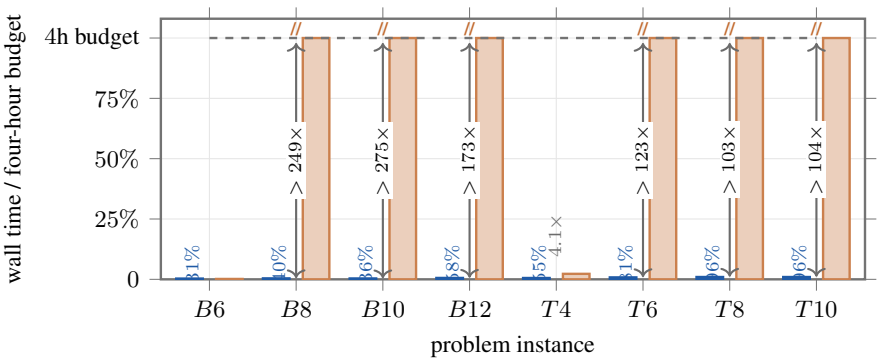


Figure 9: Online GDC wall time versus SCIP branch-and-cut wall time on the same held-out instances, normalized by the four-hour SCIP budget on a linear scale. Blue GDC bars are intentionally near the baseline; blue labels report the fraction of the four-hour budget used online. Orange SCIP bars at the budget line are time-limit cases, which provide incumbents but are not globally certified. Vertical arrows report the SCIP/GDC wall-time factor; capped SCIP cases are lower bounds.

reaches lower penalized joint merits in the reported instances. The SCIP branch-and-cut results give complementary evidence that certification for the penalized joint merit programs becomes expensive as dimension grows; except for the smallest bilevel and trilevel cases, SCIP does not certify global optimality within the four-hour limit. On the time-limit cases, GDC completes online evaluation in 52.2–138.9 seconds, using only 0.36%–0.96% of the same four-hour budget; the resulting SCIP/GDC wall-time factors are therefore conservative lower bounds, ranging from more than $103\times$ to more than $275\times$.